НИЦ «КУРЧАТОВСКИЙ ИНСТИТУТ» НИИСИ

Том 15 N 1

Труды НИИСИ

SRISA PROCEEDINGS

Москва, 2025

Федеральное государственное автономное учреждение «Федеральный научный центр
Научно-исследовательский институт системных исследований
Национального исследовательского центра «Курчатовский институт»
(НИЦ «Курчатовский институт» — НИИСИ)

ТРУДЫ НИИСИ

SRISA PROCEEDINGS

ТОМ 15 № 1

МАТЕМАТИЧЕСКОЕ И КОМПЬЮТЕРНОЕ МОДЕЛИРОВАНИЕ СЛОЖНЫХ СИСТЕМ:

ТЕОРЕТИЧЕСКИЕ И ПРИКЛАДНЫЕ АСПЕКТЫ

МОСКВА
2025

Учредитель и издатель

Федеральное государственное автономное учреждение «Федеральный научный центр Научно-исследовательский институт системных исследований Национального исследовательского центра «Курчатовский институт» (НИЦ «Курчатовский институт» — НИИСИ)

«Труды НИИСИ» — это рецензируемый научный журнал, в котором публикуются научные статьи по следующим специальностям и отраслям наук:

- 1.2.1. «Искусственный интеллект и машинное обучение» (физико-математические науки);
- 2.3.1. «Системный анализ, управление и обработка информации, статистика» (физико-математические и технические науки);
- 2.3.2. «Вычислительные системы и их элементы» (технические науки);
- 2.3.3. «Автоматизация и управление технологическими процессами и производствами» (технические науки);
- 2.3.5. «Математическое и программное обеспечение вычислительных систем, комплексов и компьютерных сетей» (физико-математические и технические науки).

Направления исследований, по которым журнал публикует оригинальные статьи определены, но не ограничены следующим перечнем: системный анализ; математика; математическое и компьютерное моделирование; задачи автоматизации и управления; обработка сигналов; компьютерное зрение и обработка изображений; распознавание образов; статистика; технологии искусственного интеллекта; информационные технологии; информационная безопасность; вычислительные системы и их элементы; микро- и наноэлектроника; высокопроизводительные вычисления; вопросы численного анализа; нейроморфные и мягкие вычисления; оптико-нейронные технологии; история науки, техники и персоналий. Журнал предназначен для научных сотрудников, инженеров и аспирантов, работающих в указанных направлениях исследований.

Миссия журнала — развитие перечисленных научных направлений в России и за рубежом, включая широкое освещение результатов исследований и обеспечение высококвалифицированных кадров печатными площадями, обеспечение высокого качества исследований путем развития механизма профессионального и общественного обсуждения научных результатов и воспитания молодого поколения ученых-исследователей.

Политика журнала ориентирована на пропаганду передовых научно-технических идей и решений в рамках развития важнейших наукоемких технологий и участия в реализации приоритетов научно-технологического развития Российской Федерации. До 2025 года журнал издавался в печатном виде с названием «Труды НИИСИ РАН» (ISSN 2225-7349).

Founder and Publisher

**Scientific Research Institute for System Analysis of the
National Research Center "Kurchatov Institute"
(NRC "Kurchatov Institute" - SRISA)**

SRISA Proceedings is a peer-reviewed journal that covers the following key areas of research:

- Artificial intelligence and machine learning
- System analysis, control, information processing, statistics
- Computing systems and their components
- Automation and control in manufacturing
- Mathematical and software support for computing systems, complexes and computer networks.

We publish original articles on topics including, but not limited to: system analysis; mathematics; computer simulation; automation and control; signal processing; computer vision and image processing; pattern recognition; statistics; artificial intelligence; information technologies; cybersecurity; computing systems and their components; micro- and nanoelectronics; high-performance computing; numerical analysis; neuromorphic and soft computing; optoneural technologies; and the history of science, technology, and researchers. Our readers include researchers, engineers, and doctoral students.

Our mission is to advance these research areas in Russia and worldwide by publishing significant results and offering leading professionals a platform to share their work. We are committed to maintaining high research standards through professional and public review while fostering the next generation of researchers.

The journal's policy is to promote advanced research and innovative solutions, foster the development of high-tech fields, and contribute to key national priorities in science and technology. Until 2025, the journal was published in print as *Trudy NIISI RAN* (Proceedings of SRISA, Russian Academy of Sciences), ISSN 2225-7349.

Andrei I. Griuntal, PhD (Phys&Math), Head of the Mathematics Department,
NRC "Kurchatov Institute" - SRISA, Moscow
Yakov M. Karandashev, PhD (Phys&Math), Leading Researcher, Center for Optic-Neural Technologies,
NRC "Kurchatov Institute" - SRISA, Moscow
Anatoly G. Kushnirenko, PhD (Phys&Math), Assoc. Prof., Head of IT for Education Department,
NRC "Kurchatov Institute" - SRISA, Moscow
Alexander N. Muranov, PhD (Engineering), Assoc. Prof., Head of the Manufacturing Automation and
Control Laboratory, NRC "Kurchatov Institute" - SRISA, Moscow
Konstantin A. Petrov, PhD (Engineering), Senior Researcher, High Performance
Microprocessor Architectures Department, NRC "Kurchatov Institute" - SRISA, Moscow
Ilya V. Semenov, PhD (Phys&Math), Leading Researcher,
Department of Computational Mathematics, NRC "Kurchatov Institute" - SRISA, Moscow
Andrey I. Tsimbalov, PhD (Engineering), Deputy Director for Microtechnology,
NRC "Kurchatov Institute" - SRISA, Moscow

Тематика номера:

Проектирование и моделирование СБИС, моделирование физических процессов в микро- и нано-электронике, информационные и компьютерные технологии, моделирование многопроцессорных систем, информационные технологии в учебной информатике, математическое моделирование и визуализация в системах виртуального окружения, моделирование когнитивных процессов.

The topic of the issue:

Design and modeling of VLSI, modeling of physical processes in micro- and nanoelectronics, information and computer technology, modeling of multiprocessor systems, information technology in educational informatics, mathematical modeling and visualization in virtual environment systems, modeling of cognitive processes.

СОДЕРЖАНИЕ

I. СИСТЕМНЫЙ АНАЛИЗ, УПРАВЛЕНИЕ И ОБРАБОТКА ИНФОРМАЦИИ, СТАТИСТИКА

И.П. Саблин, М.В. Михайлюк. Управление звуком в системах виртуального окружения 9

II. ВЫЧИСЛИТЕЛЬНЫЕ СИСТЕМЫ И ИХ ЭЛЕМЕНТЫ

С.И. Аряшев, А.А. Киселева, Е.П. Козлова, А.В. Ларионов, С.И. Ольчев, Ю.Б. Рогаткин, Л.А. Соловьева, О.В. Сысоева, И.В. Тарасов. Анализ параметров силовой схемы импульсного понижающего преобразователя напряжения для питания цифровых СБИС 14

Ю.Б. Рогаткин. 12-разрядный аналого-цифровой преобразователь конвейерного типа 21

Н.В. Масальский. Теплопроводность кремниевого полевого GAA нанотранзистора с учетом шероховатости границы 26

III. МАТЕМАТИЧЕСКОЕ И ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ, КОМПЛЕКСОВ И КОМПЬЮТЕРНЫХ СЕТЕЙ

А.А. Бурцев. Ускорение операции быстрой свёртки для множества комплексных векторов на основе технологии OpenCL 33

М.Г. Фуругян. Коррекция многопроцессорных расписаний в режиме реального времени 48

Д.И. Кадина, А.Г. Леонов, Н.С. Мартынов, К.А. Мащенко, Э.А. Орлов, А.И. Стрекалова. Автоматизация проверки на плагиат: новый подход к анализу кода в цифровой образовательной платформе Мирера 52

Н.В. Гриднев, А.С. Караваева, А.Г. Леонов, К.А. Мащенко, К.К. Пчелин, Е.Д. Тарасюк. Гибридная модель обучения: новации в учете посещаемости в цифровой образовательной платформе Мирера 58

IV. ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ И МАШИННОЕ ОБУЧЕНИЕ

В.Г. Редько. Modeling the Defense of Weak Prey Agents Against a Strong Predator Agent 65

CONTENT

I. SYSTEM ANALYSIS, CONTROL, INFORMATION PROCESSING, STATISTICS

I.P. Sablin, M.V. Mikhaylyuk. Sound Control in Virtual Environment Systems 9

II. COMPUTING SYSTEMS AND THEIR COMPONENTS

S.I. Aryashev, A.A. Kiseleva, E.P. Kozlova, A.V. Larionov, S.I. Olchev, Y.B. Rogatkin, L.A. Solovyova, O.V. Sysoeva, I.V. Tarasov. Analysis of the Parameters of the Power Circuit of a Pulse Step-Down Voltage Converter for Digital VLSI Power Supply 14

Y.B. Rogatkin. 12-bit analog-to-digital converter of conveyor type 21

N.V. Masalsky. Thermal Conductivity of a Silicon GAA Field-Effect Nanotransistor, Taking into Account the Roughness of the Boundary 26

III. MATHEMATICAL AND SOFTWARE SUPPORT FOR COMPUTING SYSTEMS, COMPLEXES AND COMPUTER NETWORKS

A.A. Burtsev. Acceleration of Fast Convolution Operation for Multiple Complex Vectors Based on OpenCL Technology 33

M.G. Furugyan. Correcting Multiprocessor Schedules in Real Time 48

D.I. Kadina, A.G. Leonov, N.S. Martynov, K.A. Mashchenko, E.A. Orlov, A.I. Strekalova. Plagiarism Detection Automation: A New Approach to Code Analysis in the Mirera Digital Educational Platform 52

N.V. Gridnev, A.S. Karavaeva, A.G. Leonov, K.A. Mashchenko, K.K. Pchelin, E.D. Tarasuk. Hybrid Learning Model: Innovations in Attendance Tracking on the Mirera Digital Educational Platform 58

IV. ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

V.G. Red'ko. Modeling the Defense of Weak Prey Agents Against a Strong Predator Agent 65

Управление звуком в системах виртуального окружения

И.П. Саблин[1*], М.В. Михайлюк[2*]

[*] НИЦ «Курчатовский институт» — НИИСИ, Москва, Российская Федерация;
[1]sablinivan97@gmail.com, [2]mix@niisi.ras.ru

Аннотация. В работе предлагается классификация источников звука в системах виртуального окружения, методы создания таких источников в системе трехмерного моделирования 3DS Max и методы управления ими с помощью виртуальных пультов и соответствующих им функциональных схем.

Ключевые слова: система виртуального окружения, источники звука, 3DS Max, виртуальные пульты управления, CtrlPanelEditor.

1. Введение

Одним из важных направлений современных исследований является разработка и усовершенствование систем виртуального окружения. Данные системы находят применение в образовании, индустрии развлечений, различных профессиональных тренажерах и т.д. Одним из главных компонентов таких систем является моделирование звука, поскольку оно повышает уровень погружения пользователя в виртуальную реальность. Однако создание реалистичной звуковой среды остается сложной задачей, требующей решения множества проблем, в частности разработки интуитивно понятных и максимально приближенных к реальным средств управления звуком. В работах [1] и [2] рассмотрено управление звуком при помощи жестов, а в работах [3] и [4] – при помощи голосовых команд. В данной статье предлагается подход к управлению источниками звука с помощью виртуальных пультов управления и их функциональных схем, которые соответствуют реальным физическим аналогам (различным микшерным пультам). Для реализации звука был разработан новый тип объекта "Источник звука" в системе трехмерного моделирования 3DS Max. Методика использования таких объектов в системе виртуального окружения VirSim, разработанной в НИЦ «Курчатовский институт» – НИИСИ, описана в работе [5]. Результаты апробации в этой системе показали адекватность предлагаемых решений и их применимость для управления источниками звука в системах виртуального окружения.

2. Типы источников звука в системах виртуального окружения

Источники звука можно разделить на два основных типа: статические и динамические.

Статические источники звука характеризуются фиксированным положением и ориентацией в пространстве на протяжении всего времени работы системы. Примерами таких источников являются неподвижные объекты (сирены, динамики, громкоговорители и др.) и некоторые природные явления (шум дождя, грозы, огня и др.).

Динамические источники звука могут менять свое положение и ориентацию. Примерами таких источников являются объекты, которые могут самостоятельно двигаться (вращающиеся сирены, динамики и др.) или прикреплены к другим подвижным объектам (гудок или громкоговоритель, расположенные на роботе). Также динамическими источниками могут быть звуки некоторых механизмов (шум движения лифта, шум работы двигателя и др.).

3. Создание объекта типа «Источник звука» в 3DS Max

Для создания объектов в виртуальных сценах используется система трехмерного моделирования 3DS Max. Данная система предоставляет стандартный набор типов объектов (геометрические, источники света, виртуальные камеры, вспомогательные объекты и др.) с соответствующими параметрами. В частности, имеется тип объекта AudioClip, который моделирует источник звука. Однако параметров AudioClip недостаточно для наших разработок, поэтому, при помощи входящего в состав 3DS Max скриптового языка MAXScript, нами создан отдельный тип «Источник звука», расширяющий параметры AudioClip. Для системы VirSim в 3DS Max уже создана новая панель инструментов, содержащая кнопки для создания расширенных или новых объектов (например, «центр масс», «двигатель», «панель управления» и др.).

На данную панель добавлена кнопка «Источник звука» (см. рис. 1) при помощи макроса macroScript (см. рис. 2). При нажатии на данную кнопку активируется специально разработанный плагин RSimSound создания объекта типа "Источник звука" и задания значений его параметров по умолчанию (см. рис. 3).

Положение и ориентация источника звука наследуются из родительского объекта AudioClip. В качестве остальных параметров в плагине задаются внутренний и внешний углы конуса затухания звука, коэффициент его затухания вне внешнего конуса, ближняя и дальняя границы затухания звука по расстоянию и коэффициент такого затухания, а также имя звукового файла и параметр зацикливания воспроизведения звука. Их можно менять, используя панель параметров «Источника звука» (см. рис. 4.).

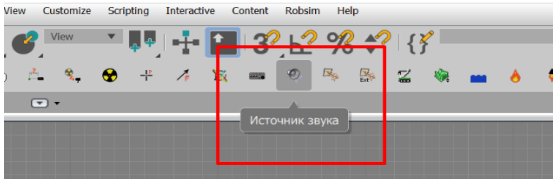


Рис. 1. Кнопка «Источник звука» на панели инструментов в 3DS Max

```
macroScript Macro_RSimSound
    category:"RobSim"
    internalcategory:"RobSim"
        tooltip:"Источник звука"
        ButtonText:"Источник звука"
        Icon:#("RobSim",20)
(
    on execute do StartObjectCreation RSimSound
    on isChecked return mcrUtils.IsCreating RSimSound
)
```

Рис. 2. Макрос создания кнопки «Источник звука»

Статические источники звука размещаются в тех же точках виртуальной сцены, где находятся их визуальные образы (например, громкоговоритель и соответствующий ему объект типа источника звука). Особым типом статических источников звука являются некоторые природные источники (например, шум дождя или града). Они не локализованы в конкретной точке сцены, однако их можно свести к статическому типу источника звука, разместив такие источники в центре виртуальной сцены, задав углы конусов 360 градусов и указав ближнюю границу затухания за границей сцены. Тогда во всех точках сцены громкость каждого из таких источников будет одинакова.

Динамические источники звука в сцене привязаны к каким-либо динамическим объектам с помощью иерархических связей и могут менять свое положение и ориентацию как самостоятельно, так и наследуя движения динамических объектов.

```
plugin Helper RSimSound
name:"Звук"
classID:#(55555,53100)
category:"RobSim"
extends:AudioClip
replaceUI:true
(
parameters paramBlock rollout:paramRollout
(
    WaveName type:#string animatable:false default:""
    InnerAngleInRadians type:#float ui:spnInnerAngleInRadians default:360.0
    OuterAngleInRadians type:#float ui:spnOuterAngleInRadians default:360.0
    OuterGain type:#float ui:spnOuterGain default:1.0
    MinGain type:#float ui:spnMinGain default:1.0
    MaxGain type:#float ui:spnMaxGain default:1.0
    RollofFactor type:#float ui:spnRollofFactor default:1.0
    PauseInMs type:#float animatable:false ui:spnTimeMs default:1.0
    loopPlaying type:#boolean animatable:false ui:chkLoop default:false
)
rollout paramRollout "Параметры"
(
button btn1 "<файл>" width:120

group "Конус направленности"
(
    label L1 "Внутренний угол [°]:" align:#left
    spinner spnInnerAngleInRadians "" range:[0,360,360] scale:1
    label L2 "Внешний угол [°]:" align:#left
    spinner spnOuterAngleInRadians "" range:[0,360,360] scale:1
    label L3 "Коэф. затухания" align:#left
    label L4 "вне конуса:" align:#left
    spinner spnOuterGain "" range:[0,1,1] scale:0.1
)

group "Затухание от расстояния"
(
    label L5 "Ближняя граница [см]" align:#left
    spinner spnMinGain "" range:[100,1000000, 1] scale:10
    label L6 "Дальняя граница [см]" align:#left
    spinner spnMaxGain "" range:[100,1000000, 1] scale:10
    label L "Коэф. скорости затухания" align:#left
    spinner spnRollofFactor "" range:[0.1,1,1] scale:0.1
)
group "Периодичность"
(
    checkbox chkLoop "Зациклить" width:120
```

Рис. 3. Фрагмент программного кода плагина RSimSound на скриптовом языке MAXScript

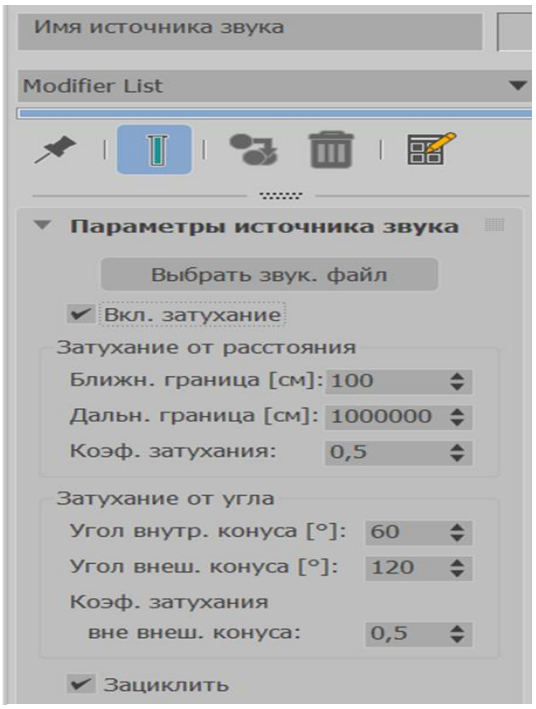


Рис. 4. Параметры источника звука

Для сохранения на диске всех объектов сцены используется макрос macroScript Export_Scene. В частности, параметры всех источников звука, заданные при помощи плагина RsimSound, сохраняются в бинарном файле с расширением sls. Для этого разработана функция WriteSoundParams, при помощи которой производится поиск объектов типа RSimSound среди всех объектов типа Helper в сцене и выполняется их запись в файл.

4. Подготовка аудиофайлов

Подготовка аудиофайлов для интеграции в виртуальную среду выполнялась с помощью звуковых материалов из открытых библиотек (Freesound, Zvukogram и аналогичных) и программы Audacity обработки аудио. Все исходные записи независимо от их первоначального формата приводились к единому стандарту - монофонический звук с частотой дискретизации 44.1 кГц и разрядностью 16 бит. Выбранные характеристики позволяют избежать потерь качества при цифровой обработке звука и обеспечивают достаточный запас для последующей трансформации звука в виртуальной среде (пространственной обработки, изменения тональности, громкости и т.д.). Для каждого файла проводилось удаление фоновых шумов, нормализация громкости, точная обрезка по времени и др.

5. Управление звуком в системе виртуального окружения

С точки зрения управления, используемые в VirSim источники звука можно разделить на управляемые (сирена, звук двигателя, град и др.) и неуправляемые (звуки столкновений объектов, звуки трения объектов при движении и т.д.). Для управления используются виртуальные пульты, созданные в разработанном ранее редакторе CtrlPanelEditor [6]. Данный редактор состоит из двух рабочих окон.

В первом окне можно создавать виртуальные пульты управления, а во втором - функциональные схемы этих пультов. Виртуальные пульты состоят из элементов управления (кнопки, ползунки, переключатели и т.д.). Пример виртуального пульта управления источником звука (сиреной) проиллюстрирован на рис. 5.

После добавления элементов управления в окно виртуального пульта они автоматически добавляются и в окна функциональных схем в виде блоков элементов управления. Например, тумблеру ВКЛ/ВЫКЛ на рис. 5 будет соответствовать блок Switch на рис. 6. В функциональных схемах выходы блоков можно соединять с входами блоков других типов (логических, арифметических и др.) и блоками исполнительных

Рис. 5. Виртуальный пульт управления

устройств (двигателей роботов, манипуляторов и др.). Пример функциональной схемы управления источником звука приведен на рис. 6.

Источник звука (на рис. 6 изображен в виде динамика с именем Siren) также является блоком исполнительных устройств и имеет три входа (на рисунке сверху вниз) - «Включение/Выключение» для активации/деактивации звука, «Пауза» для временной остановки воспроизведения и «Громкость» для регулировки уровня звука.

Некоторые управляемые источники звука связаны с событиями. Например, звук работы двигателя запускается (заглушается) при включении (выключении) двигателя. В этом случае при начале движения робота, в функциональной схеме пульта управления этим роботом, от выхода «Датчик положения» блока «Двигатель» подается сигнал на вход «Включение» блока «Источника звука». Звук работы двигателя будет воспроизводиться до тех пор, пока оператор не выключит двигатель.

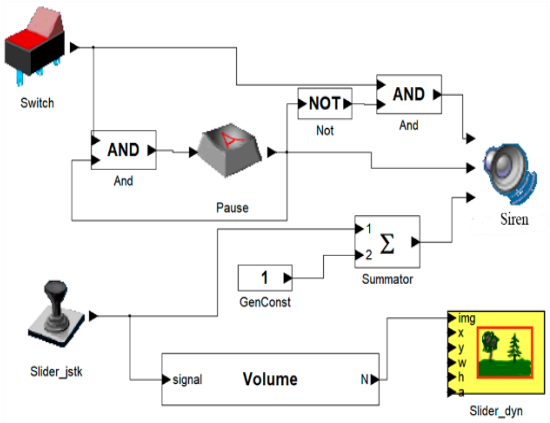


Рис. 6. Функциональная схема управления источником звука

6. Управление громкостью источника звука в системе VirSim

Громкость доходящего до слушателя звука зависит от громкости самого источника звука и степени затухания звука на пути к слушателю. Затухание звука зависит от изменения взаимного расстояния и ориентации слушателя и источника звука. Подробнее это описано в статье [5].

В системе VirSim изменение громкости источников звука реализовано тремя различными методами.

В первом методе управление осуществляется через индивидуальные виртуальные пульты управления (как на рис. 5), где статические и некоторые динамические источники имеют собственный регулятор громкости. Эти пульты позволяют оператору вручную изменять уровень звука от 0 до 1. В частности, интенсивность природных явлений (например, уровень силы дождя) регулируется предустановленными режимами, где каждый вариант соответствует определенному уровню громкости (от 0 до 1).

Второй метод основан на изменении громкости источников звука, зависящем от изменения скорости этих источников. Например, для роботов реализовано уменьшение или увеличение громкости звука их двигателей, связанное с их скоростью движения. В функциональной схеме каждого робота происходит расчет громкости двигателя этого робота пропорционально его текущей скорости, где максимальное значение скорости соответствует уровню громкости 1, а нулевая скорость — уровню громкости 0. При этом используется линейная интерполяция между уровнями (0 и 1 громкости) для исключения резких скачков аудио сигнала.

Третий метод — это изменение громкости источника звука с использованием нескольких различных аудиофайлов. Например, уменьшение громкости звука огня реализовано через переключение между двумя аудиофайлами. Пока огонь горит, воспроизводится основной файл с постоянной громкостью, а при тушении этого огня система автоматически запускает второй файл со звуком огня при тушении. Длительность второго файла связана с визуальным затуханием пламени, что создаёт эффект постепенного исчезновения огня как визуально, так и на слух, без резких обрывов звука. Для звука движения промышленного лифта реализована аналогичная система звукового сопровождения. Особенность реализации заключается в использовании разных звуков для разных состояний – равномерный промышленный шум для фазы движения и звон тросов для фазы остановки, что позволяет добиться соответствия звука реальной работе лифтовых систем. Такое решение обеспечивает плавные переходы между состояниями без необходимости программной обработки громкости и сохраняет натуральность звучания за счет использования специально записанных аудиодорожек для каждого режима работы.

7. Заключение

В данной работе предлагается классификация источников звука в системах виртуального окружения, методы создания этих источников звука в системе трехмерного моделирования 3DS Max и методы управления источниками звуков в системах виртуального окружения.

Публикация выполнена в рамках государственного задания НИЦ «Курчатовский институт» — НИИСИ по теме № FNEF-2024-0002 «Математическое моделирование многомасштабных динамических процессов и системы виртуального окружения».

Sound Control in Virtual Environment Systems

I.P. Sablin, M.V. Mikhaylyuk

Abstract. The paper proposes a classification of sound sources in virtual environment systems, methods for creating such sources in the 3DS Max three-dimensional modeling system, and methods for controlling them using virtual control panels and their corresponding functional circuits.

Keywords: Virtual environment system, sound sources, 3DS Max, virtual control panels, CtrlPanelEditor.

Литература

1. Mahya Khazaei. Real-Time Gesture-Based Sound Control System. A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of Master of science. Iran University of Science and Technology. (2018).

2. Lokesh S. Khedekar, Manas Patil. Gesture-Based Volume Control Using Computer Vision and Audio

Processing. International Conference on Communication and Signal Processing (ICCSP). (2024). DOI: 10.1109/ICCSP60870.2024.10543441

3. Jan Hombeck, Henrik Voigt, Timo Heggemann, Rr Datta. Tell Me Where To Go: Voice-Controlled Hands-Free Locomotion for Virtual Reality Systems. IEEE Conference Virtual Reality and 3D User Interfaces (VR). (2023). DOI: 10.1109/VR55154.2023.00028.

4. Hunter Osking, John A. Doucette. Enhancing Emotional Effectiveness of Virtual-Reality Experiences with Voice Control Interfaces. Immersive Learning Research Network. (2019), 199-209. DOI:10.1007/978-3-030-23089-0_15

5. Саблин И.П., Михайлюк М.В., Омельченко Д.В., Кононов Д.А. Моделирование звука в системах виртуального окружения. Успехи кибернетики. Т. 6, (2025). № 2, 92–99.

6. М.В. Михайлюк, М.А. Торгашев. Визуальный редактор и модуль расчета функциональных схем для имитационно-тренажерных комплексов. Программные продукты и системы, № 4 (108), (2014), 10-15. DOI: 10.15827/0236-235X.108.010-015

Анализ параметров силовой схемы импульсного понижающего преобразователя напряжения для питания цифровых СБИС

С.И. Аряшев[1*], А.А. Киселева[2*], Е.П. Козлова[3*], А.В. Ларионов[4*], С.И. Ольчев[5*], Ю.Б. Рогаткин[6*], Л.А. Соловьева[7*], О.В. Сысоева[8*], И.В. Тарасов[9*]

[*] НИЦ «Курчатовский институт» — НИИСИ, Москва, Российская Федерация;

[1] aserg@cs.niisi.ras.ru; [2] kiseleva@cs.niisi.ras.ru; [3]nikolaeva@cs.niisi.ras.ru; [4]alar@cs.niisi.ras.ru; [5]olchev@cs.niisi.ras.ru; [6]ryb@cs.niisi.ras.ru; [7]isolov@cs.niisi.ras.ru; [8]olga@cs.niisi.ras.ru; [9]tarasov@cs.niisi.ras.ru

Аннотация. Разработка импульсного понижающего преобразователя напряжения начинается с расчёта параметров дросселя и выходного конденсатора. Величина индуктивности дросселя и ёмкости выходного конденсатора зависят от множества характеристик преобразователя, таких, как мощность преобразователя, величины входного и выходного напряжения, допустимый уровень пульсации выходного напряжения, допустимая относительная пульсация выходного тока, частота модуляции. Как правило, характеристики преобразуемого напряжения и мощность преобразователя задаются техническим заданием и не могут быть выбранными разработчиком самостоятельно. Частота же модуляции не относится к параметрам, характеризующим выходное питание преобразователя, и может быть выбрана разработчиком на его усмотрение. Величины индуктивности дросселя и ёмкости выходного конденсатора имеют существенную зависимость от выбранной частоты модуляции. Чем выше частота модуляции, тем меньше величина индуктивности и ёмкости выходного конденсатора преобразователя. Это позволяет минимизировать общие физические размеры импульсного преобразователя. Однако при увеличении частоты модуляции увеличиваются потери энергии на силовом ключе и усиливается влияние электромагнитных помех. Выбор частоты модуляции импульсного преобразователя напряжения — это всегда поиск компромисса между минимизацией размеров дросселя с выходным конденсатором и снижением КПД преобразователя вместе со снижением помехозащищённости питаемого устройства. В данной статье представлен пример анализа основных параметров силовой схемы преобразователя в широком диапазоне частот модуляции. Эти расчёты помогают разработчику найти оптимальное соотношение частоты модуляции с величинами дросселя и выходного конденсатора, если в соответствии с техническим заданием допускается использование различных частот модуляции. Также представлен пример расчёта зависимости параметров импульсного преобразователя от выходного напряжения и выходной мощности при фиксированной частоте модуляции.

Ключевые слова: импульсный понижающий преобразователь напряжения, buck-конвертер, частота модуляции, цифровые СБИС.

1. Введение

Снижение проектных норм и повышение производительности современных СБИС ставят перед разработчиками специфические требования к источникам питания для этих устройств. Рост производительности и скорости обработки данных специализированных заказных интегральных схем (ASIC), цифровых сигнальных процессоров (DSP - процессоры), программируемых вентильных матриц (FPGA) и других цифровых устройств приводят к значительному увеличению потребляемого тока. При этом, с уменьшением проектных норм для избегания влияния электромиграции или пробоя перехода транзисторов используется более низкое напряжение питания. Мощность потребления этих цифровых устройств может существенно изменяться в зависимости от режима функционирования. Смена режима работы цифрового устройства между высокой и низкой производительностью сопровождается существенными переходными процессами в токе нагрузки, что может негативно сказаться на стабильности напряжения питания. Таким образом, источники питания современных СБИС должны поддерживать низкие напряжения питания при высоких значениях токов нагрузки, обеспечивая при этом стабильность выдаваемого напряжения от переходных процессов при изменении режимов работы питаемых цифровых устройств.

Наиболее подходящим элементом питания, удовлетворяющим этим требованиям, является импульсный понижающий преобразователь. Этот тип преобразователей характеризуется высоким КПД и возможностью быстрой коррекции выдаваемой мощности в зависимости от измене-

ния тока нагрузки. Различные цифровые устройства в составе СБИС могут иметь различное напряжение питания. Одним из эффективных способов применения импульсных преобразователей является расположение источников питания непосредственно рядом с нагрузкой [1]. Локальное расположение понижающих преобразователей позволяет снизить влияние распределительной сети и сократить длину проводников печатной платы. Уменьшение паразитных параметров соединительных проводников между источником питания и нагрузкой способствует увеличению быстродействия компенсационной обратной связи импульсного преобразователя при стремительном изменении тока нагрузки. Благодаря этому улучшается стабильность выдаваемого напряжения питания.

Эти преимущества локального расположения понижающих преобразователей нашли широкое распространения в распределённой системе питания бортовой аппаратуры космических аппаратов [2]. Понижающие DC/DC преобразователи, расположенные в непосредственной близости от питаемого функционального узла, называются POL (point of load). Преобразователи POL, применяемые в бортовой аппаратуре космических аппаратов, обладают повышенной радиационной стойкостью, высоким КПД и низкой рассеиваемой мощностью, что является особенно актуальным в условиях ограниченного теплоотвода.

Разработка импульсного понижающего преобразователя всегда сопряжена с поиском компромисса между частотой модуляции и размерами дросселя с выходным конденсатором. Увеличение частоты модуляции позволяет снизить размеры дросселя и выходного конденсатора. Однако вместе с этим вырастает потребляемая мощность и увеличивается влияние электромагнитных помех на соседние устройства. Каждое удвоение частоты модуляции импульсного преобразователя снижает его КПД примерно на 2%. Поэтому максимальная частота модуляции импульсных преобразователей обычно не превышает 1.6 МГц [3]. С целью снижения влияния электромагнитных помех от импульсных преобразователей разработчикам необходимо соблюдать определённые требования при проектировании печатной платы, например, избегать резких изгибов металлических проводников и Т–образных ответвлений [4]. В импульсных преобразователях POL для электронной аппаратуры космических аппаратов с целью минимизации влияния электромагнитных помех на питаемые устройства, а также с целью минимизации рассеиваемого тепла частота модуляции как правило составляет не более 400...500 кГц [2], [5].

2. Схема импульсного понижающего преобразователя напряжения

Импульсные понижающие преобразователи передают со входа на выход небольшие порции энергии, используя ключ, диод, индуктивность и конденсатор. Наиболее распространённой схемой такого преобразователя является так называемый buck-конвертер. Если отсутствует необходимость в гальванической развязке между источником входного напряжения и выходом, то схема buck-конвертера выглядит так, как представлена на рис. 1. В схемах с гальванической развязкой используется дроссель с двумя магнитосвязанными катушками.

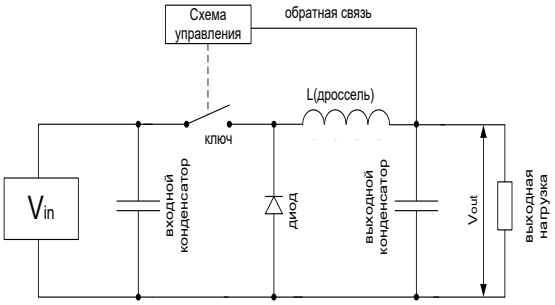


Рис. 1. Функциональная схема импульсного понижающего преобразователя напряжения buck-конвертера

Импульсный преобразователь состоит из входного источника напряжения, стабилизирующего входного конденсатора, ключа, диода, дросселя, выходного конденсатора и схемы управления с обратной связью. Цикл работы импульсного преобразователя состоит из двух этапов. На первом этапе ключ замкнут, диод закрыт, ток от входного источника протекает через дроссель, выходную нагрузку и выходной конденсатор. Порция энергии от входного источника напряжения аккумулируется за счёт тока дросселя и заряда на выходном конденсаторе. Выходной конденсатор также необходим для сглаживания пульсаций.

Схема управления через обратную связь непрерывно сравнивает выходное напряжение V_{out} с заданным эталонным значением. Как только напряжение на выходном конденсаторе достигает этого значения, начинается второй этап работы преобразователя. Схема управления размыкает ключ, тем самым отключая диод, дроссель, выходной конденсатор и выходную нагрузку от входного источника напряжения. Резкий скачок напряжения на входе дросселя отпирает диод и ток через катушку продолжает протекать уже через диод.

В импульсных преобразователях напряжения управление силовым ключом возможно двумя

способами. Широтно-импульсной модуляцией (ШИМ) и частото-импульсной модуляцией (ЧИМ). Управление с помощью ШИМ используют в преобразователях, которые питают нагрузку с малоизменяющимся током потребления. В таких схемах частота переключения ключа f_{SW} постоянна, а порции энергии от входного источника регулируются продолжительностью импульса. Далее будут приведены расчёты для преобразователя со схемой управления с ШИМ.

3. Расчёт индуктивности дросселя и ёмкости выходного конденсатора импульсного преобразователя

Проектирование импульсного преобразователя напряжения начинается с расчёта парамеров силовой схемы. Это прежде всего индуктивность дросселя L и ёмкость выходного конденсатора C_{out}. Исходные данные для расчёта следующие:

f_{sw} – частота переключения ключа (частота модуляции);

V_{in_max} – максимальное входное напряжение;

V_{out} – выходное напряжение импульсного преобразователя;

ΔV – максимально допустимый всплеск выходного напряжения;

I_{out} – выходной ток;

LIR – допустимая относительная пульсация выходного тока. Рассчитывается по формуле (1).

$$LIR = \frac{I_{out\_max} - I_{out\_min}}{I_{out}} \quad (1)$$

где I_{out_min} – минимальный выходной ток, I_{out_max} – максимальный выходной ток. При этом выходной ток $I_{out} = (I_{out\_max} + I_{out\_min})/2$. Как правило, параметр LIR принимают равным 0,3…0,4.

Индуктивность дросселя рассчитывается по формуле (2).

$$L = \frac{(1-\frac{V_{out}}{V_{in\_max}})\cdot V_{out}}{LIR\cdot I_{out}\cdot(\frac{LIR}{2}+1)\cdot f_{sw}} \quad (2)$$

Для более точной корректировки значения индуктивности дросселя учитывают среднее падение напряжения на диоде и среднее падение напряжения на ключе (3).

$$L = \frac{(1-\frac{(V_{out}+V_f)}{(V_{in\_max}+V_f-V_{sat})})\cdot(V_{out}+V_f)}{LIR\cdot I_{out\_max}\cdot f_{sw}} \quad (3)$$

где V_f – среднее падение напряжения на диоде, V_{sat} – среднее падение напряжения на ключе.

Существующие подходы расчёта параметров силовой схемы импульсного понижающего преобразователя как правило основаны на использовании определённой фиксированной частоты модуляции, выбранной разработчиком [6].

На рис. 2. представлена зависимость индуктивности дросселя L от частоты модуляции f_{sw}.

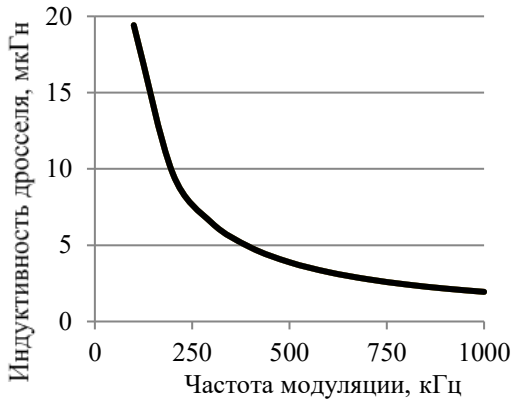


Рис. 2. Зависимость индуктивности дросселя импульсного преобразователя напряжения от частоты модуляции

В качестве примера расчёта параметров силовой схемы импульсного понижающего преобразователя приняты следующие исходные данные: $V_{in\_max} = 3$ В, $V_{out} = 1$ В, $I_{out} = 1$ А, $\Delta V = 0{,}1$ В, LIR = 0,3. Как видно из графика, зависимость индуктивности дросселя обратно пропорциональна частоте модуляции. Для частоты в районе 1 МГц требуемая индуктивность при заданных параметрах составляет около 2 мкГн. Также стоит отметить, что дроссель должен быть подобран так, чтобы максимально допустимый пиковый ток I_{out_max} был не больше тока насыщения дросселя $I_{нас}$. Т.е. габариты и материал сердечника должны быть такими, чтобы, намотав на этом сердечнике требуемую индуктивность, ток насыщения катушки был больше максимального пикового тока.

Расчёт ёмкости выходного конденсатора производится по формуле (4).

$$C''_{out} = \frac{I_{out\_max}}{8\cdot f_{sw}\cdot \Delta V} \quad (4)$$

Эта формула позволяет определить минимальную величину выходного конденсатора, при котором изменение выходного напряжения V_{out} будет находиться в допустимом диапазоне ΔV при условии, что выходной ток I_{out} находится в диапазоне от I_{out_max} до I_{out_min}. Однако возможен наихудший случай, когда при максимальном

токе через катушку (т.е. при пиковом токе) выходной ток скачком меняется от $I_{out\ max}$ до нуля. Для этого наихудшего случая выходная ёмкость должна забрать на себя излишек заряда и не допустить превышение выходного напряжения на величину ΔV. Формула для расчёта выходной ёмкости для наихудшего случая имеет следующий вид (5):

$$C_{out} = \frac{L \cdot I_{out}^2 \cdot (1+\frac{LIR}{2})^2}{(V_{out}+\Delta V)^2 - V_{out}^2} \quad (5)$$

На рис. 3. приведены два графика зависимости выходной ёмкости от частоты модуляции. Для базового режима работы импульсного преобразователя и для наихудшего случая при максимальном всплеске тока дросселя. Исходные данные для построения графика заданы те же: V_{in_max} = 3 В, V_{out} = 1 В, I_{out} = 1 А, ΔV = 0,1 В, LIR = 0,3.

Как видно из графиков на рис. 3, для малых частот модуляции разница в значениях выходной ёмкости для базового и для наихудшего случая может составлять более 100 мкФ. В относительном выражении, ёмкость для наихудшего случая должна быть примерно в 10 раз больше, чем для штатного режима работы импульсного преобразователя.

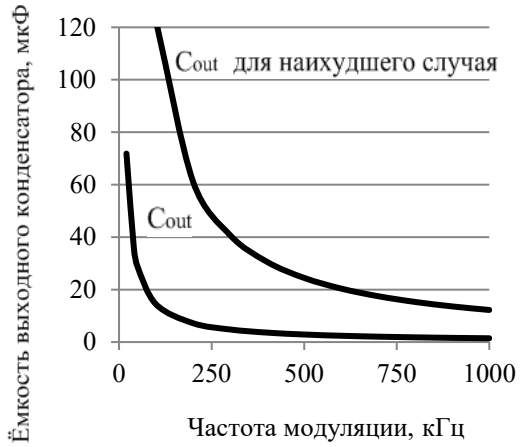


Рис. 3. Зависимость ёмкости выходного конденсатора импульсного преобразователя напряжения от частоты модуляции

4. Частотный анализ LC-фильтра силовой схемы

Для нормальной работы импульсного понижающего преобразователя частота модуляции f_{SW} должна быть значительно выше частоты среза LC-фильтра f_{LC}. В противном случае фильтрация импульсного сигнала будет неэффективной, и на выходе будут наблюдаться значительные пульсации. Частота среза LC- фильтра определяется по формуле (6).

$$f_{LC} = \frac{1}{2 \cdot \pi \cdot \sqrt{L \cdot C_{out}}} \quad (6)$$

Параметры понижающего импульсного преобразователя рекомендуется подбирать таким образом, чтобы частота модуляции f_{SW} была в 10…50 раз больше частоты среза LC- фильтра.

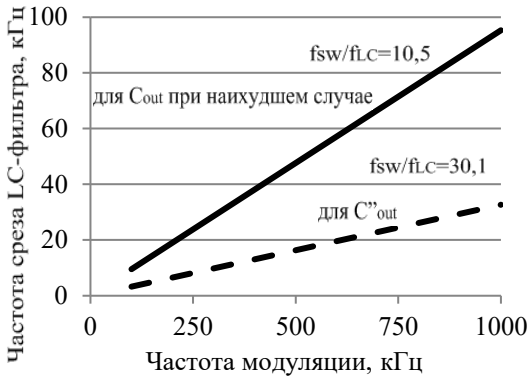


Рис. 4. Зависимость частоты среза LC- фильтра импульсного преобразователя от частоты модуляции для C_{out} при наихудшем случае (сплошная линия) и для $C"_{out}$ при базовой работе преобразователя (пунктирная линия).

На рис. 4. Представлены зависимости частоты среза LC-фильтра импульсного преобразователя от частоты модуляции для двух схем: с выходной ёмкостью C_{out} при наихудшем случае (сплошная линия) и для выходной ёмкости $C"_{out}$ при базовой работе преобразователя (пунктирная линия). Также для каждого графика представлены отношение частоты модуляции f_{SW} к частоте среза LC- фильтра f_{LC}. Расчёты приведены для V_{in_max} = 3 В, V_{out} = 1 В, I_{out} = 1 А, ΔV = 0,1 В, LIR = 0,3.

Как видно из рисунка, при обоих значениях емкостей (для базового и для наихудшего случая) частота среза LC- фильтра в 10,5 раза и в 30,1 раза меньше частоты модуляции соответственно, что удовлетворяет условию стабильной работы понижающего преобразователя.

5. Пример расчёта зависимости параметров импульсного преобразователя от выходного напряжения и выходной мощности при частоте модуляции 1 МГц

В таблицах 1, 2 и 3 представлены расчётные значения индуктивности дросселя L, ёмкости выходного конденсатора C_{out} (для наихудшего

случая) и частота среза LC-фильтра f_{LC} в зависимости от потребляемой мощности нагрузки P_{out} для разных значений выходного напряжения V_{out}. В таблице 1 представлены данные для V_{out} = 0,5 В, в таблице 2 представлены данные для V_{out} = 1 В и в таблице 3 представлены данные для V_{out} = 2 В. Все расчёты проделаны при фиксированных параметрах V_{in_max} = 3 В, ΔV = 0,1 В, LIR = 0,3 и частоты модуляции f_{SW} = 1МГц.

Таблица 1. Расчётные данные преобразователя для V_{out} = 0,5 В

P_{out}, Вт	$1*10^{-6}$	$1*10^{-5}$	$1*10^{-4}$	$1*10^{-3}$	$1*10^{-2}$	$1*10^{-1}$	1	5	10
L, Гн	603 $*10^{-3}$	603 $*10^{-4}$	603 $*10^{-5}$	603 $*10^{-6}$	603 $*10^{-7}$	603 $*10^{-8}$	603 $*10^{-9}$	121 $*10^{-9}$	60,3 $*10^{-9}$
C_{out}, Ф	2,90 $*10^{-11}$	2,90 $*10^{-10}$	2,90 $*10^{-9}$	2,90 $*10^{-8}$	2,90 $*10^{-7}$	2,90 $*10^{-6}$	2,90 $*10^{-5}$	1,45 $*10^{-4}$	2,90 $*10^{-4}$
I_{out}, А	$2*10^{-6}$	$2*10^{-5}$	$2*10^{-4}$	$2*10^{-3}$	$2*10^{-2}$	$2*10^{-1}$	2	10	20
I_{out_max}, А	2,3 $*10^{-6}$	2,3 $*10^{-5}$	2,3 $*10^{-4}$	2,3 $*10^{-3}$	2,3 $*10^{-2}$	2,3 $*10^{-1}$	2,3	11,5	23

Примечание: расчётная частота среза LC- фильтра f_{LC} составляет 38 кГц.

Таблица 2. Расчётные данные преобразователя для V_{out} = 1 В

P_{out}, Вт	$1*10^{-6}$	$1*10^{-5}$	$1*10^{-4}$	$1*10^{-3}$	$1*10^{-2}$	$1*10^{-1}$	1	5	10
L, Гн	1930* $*10^{-3}$	1930* $*10^{-4}$	1930* $*10^{-5}$	1930* $*10^{-6}$	1930* $*10^{-7}$	1930* $*10^{-8}$	1930* $*10^{-9}$	387* $*10^{-9}$	193* $*10^{-9}$
C_{out}, Ф	$1,21*10^{-11}$	$1,2*10^{-10}$	$1,21*10^{-9}$	$1,21*10^{-8}$	$1,21*10^{-7}$	$1,21*10^{-6}$	$1,21*10^{5}$	$0,61*10^{-4}$	$1,21*10^{-4}$
I_{out}, А	$1*10^{-6}$	$1*10^{-5}$	$1*10^{-4}$	$1*10^{-3}$	$1*10^{-2}$	$1*10^{-1}$	1	5	10
I_{out_max}, А	1,15 $*10^{6}$	1,15 $*10^{5}$	1,15 $*10^{4}$	1,15 $*10^{3}$	1,15 $*10^{2}$	1,15 $*10^{1}$	1,15	5,75	11,5

Примечание: расчётная частота среза LC- фильтра f_{LC} составляет 32,8 кГц.

Таблица 3. Расчётные данные преобразователя для V_{out} = 2 В

P_{out}, Вт	$1*10^{-6}$	$1*10^{-5}$	$1*10^{-4}$	$1*10^{-3}$	$1*10^{-2}$	$1*10^{-1}$	1	5	10
L, Гн	3870* $*10^{-3}$	3870* $*10^{-4}$	3870* $*10^{-5}$	3870* $*10^{-6}$	3870* $*10^{-7}$	3870* $*10^{-8}$	3870* $*10^{-9}$	773* $*10^{-9}$	387* $*10^{-9}$
C_{out}, Ф	0,312* $*10^{-11}$	0,312* $*10^{-10}$	0,312* $*10^{-9}$	0,312* $*10^{-8}$	0,312* $*10^{-7}$	0,312* $*10^{-6}$	0,312* $*10^{-5}$	0,156* $*10^{-4}$	0,312* $*10^{-4}$
I_{out}, А	0,5 $*10^{-6}$	0,5 $*10^{-5}$	0,5 $*10^{-4}$	0,5 $*10^{-3}$	0,5 $*10^{-2}$	0,5 $*10^{-1}$	0,5	2,5	5
I_{out_max}, А	0,575* $*10^{-6}$	0,575* $*10^{-5}$	0,575* $*10^{-4}$	0,575* $*10^{-3}$	0,575* $*10^{-2}$	0,575* $*10^{-1}$	0,575	2,875	5,75

Примечание: расчётная частота среза LC- фильтра f_{LC} составляет 45,9 кГц.

Как видно из представленных данных, частота среза LC- фильтра f_{LC} во всём диапазоне выходных напряжений 0,5…2 В не менее, чем в 20 раз меньше по сравнению с частотой модуляции f_{SW} = 1 МГц, что удовлетворяет условию стабильной работы преобразователя, при котором частота модуляции f_{SW} должна быть в 10…50 раз больше частоты среза LC- фильтра . С увеличением мощности нагрузки возрастает величина ёмкости выходного конденсатора. Вместе с тем, уменьшается индуктивность дросселя, но возрастает ток, который должен выдержать дроссель. Расчёт максимального тока через дроссель позволяет оценить требуемое значение по току насыщения дросселя.

6. Заключение

Анализ параметров силовой схемы импульсного понижающего преобразователя напряжения показал существенную зависимость индуктивности дросселя и ёмкости выходного конденсатора от частоты модуляции.

Величина индуктивности дросселя и выходного конденсатора обратно пропорциональны частоте модуляции преобразователя. На основе расчётов было установлено, что для стабилизации выходного напряжения импульсного преобразователя в заданном диапазоне при максимальном всплеске тока дросселя ёмкость выходного дросселя должна иметь примерно в 10 раз большую величину, чем при штатном режиме работы преобразователя. Пример расчёта параметров импульсного преобразователя для широкого диапазона выходной мощности показал существенное увеличение величины выходной ёмкости выходного конденсатора и уменьшение величины индуктивности дросселя при росте выходной мощности преобразователя. Этот расчёт позволяет произвести оценку тока насыщения дросселя для заданной выходной мощности. Публикация выполнена в рамках государственного задания НИЦ «Курчатовский институт» — НИИСИ по теме № FNEF-2024-0003.

Analysis of the Parameters of the Power Circuit of a Pulse Step-Down Voltage Converter for Digital VLSI Power Supply

S.I. Aryashev, A.A. Kiseleva, E.P. Kozlova, A.V. Larionov, S.I. Olchev, Y.B. Rogatkin, L.A. Solovyova, O.V. Sysoeva, I.V. Tarasov

Abstract. The development of a pulse step-down voltage converter begins with the calculation of the parameters of the inductor and the output capacitor. The value of the inductance of the inductor and the capacitance of the output capacitor depend on many characteristics of the converter, such as the power of the converter, the values of the input and output voltage, the permissible level of output voltage ripple, the permissible relative pulse of the output current, and the modulation frequency. As a rule, the characteristics of the converted voltage and the power of the converter are set by the technical specification and cannot be independently selected by the developer. The modulation frequency is not one of the parameters characterizing the output power of the converter and can be selected by the developer at his discretion. The values of the inductance of the inductor and the capacitance of the output capacitor have a significant dependence on the selected modulation frequency. The higher the modulation frequency, the lower the inductance and capacitance of the output capacitor of the converter. This minimizes the overall physical dimensions of the pulse converter. However, as the modulation frequency increases, energy losses on the power key increase and the effect of electromagnetic interference increases. Choosing the modulation frequency of a pulse voltage converter is always a search for a compromise between minimizing the size of the inductor with the output capacitor and reducing the efficiency of the converter, along with reducing the noise immunity of the powered device. This article provides an example of the analysis of the main parameters of the converter's power circuit in a wide range of modulation frequencies. These calculations help the developer to find the optimal ratio of the modulation frequency with the values of the inductor and the output capacitor, if different modulation frequencies are allowed in accordance with the technical specification. An example of calculating the dependence of the parameters of a pulse converter on the output voltage and output power at a fixed modulation frequency is also presented.

Keywords: pulse step-down voltage converter, buck converter, modulation frequency, digital VLSI.

Литература

1. А. В. Лукин. Новые направления развития преобразователей постоянного напряжения (по материалам зарубежной печати). «Электропитание», 2011, № 2, 2-4.

2. В. Жданкин. Радиационно-стойкие низковольтные DC/DC-преобразователи для распределенных систем электропитания ракетно-космической техники. «Компоненты и технологии», 2011, № 7, 130-136.

3. К. Мараско. Эффективное применение понижающих преобразователей постоянного тока производства компании Analog Devices. «Компоненты и технологии», 2011, №10, 55 – 58.

4. П. Ильин. Проблемы электромагнитной совместимости импульсных источников питания. «Электронные компоненты», 2010, №12, 11-16.

5. В. Жданкин. Высокоэффективные радиационно-стойкие DC/DC-преобразователи с низковольтными выходами — оптимальное решение для современных цифровых нагрузок. «Компоненты и технологии», 2011, № 10, 130-136.

6. Д. Шелле, Д. Касторена. Советы по проектированию понижающих преобразователей. «Новости Электроники», 2007, №8, 23-28.

12-разрядный аналого-цифровой преобразователь конвейерного типа

Ю.Б. Рогаткин

НИЦ «Курчатовский институт» — НИИСИ, Москва, Российская Федерация;

ryb@cs.niisi.ras.ru;

Аннотация. Представлен КМОП 12-разрядный аналого-цифровой преобразователь конвейерного типа с проектными нормами 28 нм. Сравнительно низкое энергопотребление при достаточно высокой частоте дискретизации до 125 МГц достигается за счет использования в конвейерной архитектуре полностью дифференциальных двухкаскадных операционных усилителей, работающих в конвейерной архитектуре АЦП 1,5 бита на каскад с напряжением питания 1,8В.

Ключевые слова: аналого-цифровой преобразователь, частота дискретизации, разрядность.

1. Введение

Поскольку сигналы реального времени всегда являются аналоговыми по своей природе, существует необходимость преобразования этих аналоговых величин в цифровой сигнал, соответствующий любому применению. Аналого-цифровой преобразователь (АЦП) — это устройство, которое преобразует аналоговый сигнал, такой как напряжение или ток, в цифровой сигнал (т.е. представляет его в двоичном виде - 0 и 1). Таким образом, преобразование аналогового сигнала в цифровой обычно выполняется с помощью преобразователей данных. Аналого-цифровой и цифроаналоговый преобразователь (ЦАП) — это два очевидных типа преобразователей данных, каждый из которых применяется в соответствии с требуемым применением. При проектировании сенсорной сети, АЦП играют очень важную роль. Сигналы, получаемые от датчика, дискретизируются, квантуются и кодируются, а эквивалентный цифровой сигнал получается с помощью стандартного аналого-цифрового преобразователя. Как правило, чтобы избежать наложения, дискретизация входного сигнала должна выполняться со скоростью, более чем в два раза превышающей полосу пропускания входного аналогового сигнала. Однако, бывают применения, когда АЦП работает в режиме субдискретизации, когда спектр входного сигнала лежит в полосе, соответствующей более высокой зоне Найквиста, чем первая. В этом случае важную роль играет полоса пропускания АЦП (Analog Bandwidth) [1]. По сравнению с другими АЦП, в конвейерных АЦП используется меньшее количество компараторов, что приводит к меньшей задержке. В портативных устройствах передачи данных и системах беспроводной связи конвейерный АЦП является важным компонентом, поскольку он может выдавать выходные данные с высокой скоростью и разрешением. Чтобы обеспечить наилучшее соотношение между скоростью и разрешением для достижения оптимальных условий, используется конвейерный АЦП. Это приводит к достижению идеального размера микросхемы и относительно небольшому рассеиванию мощности [2].

2. Архитектура и конструкция АЦП

Одна из простейших реализаций конвейерных АЦП, включающих цифровую коррекцию (резервирование), основана на архитектуре с производительностью 1,5 бита на каскад, показанной на рис. 1. Эта архитектура традиционно широко используется для увеличения скорости преобразования [3].

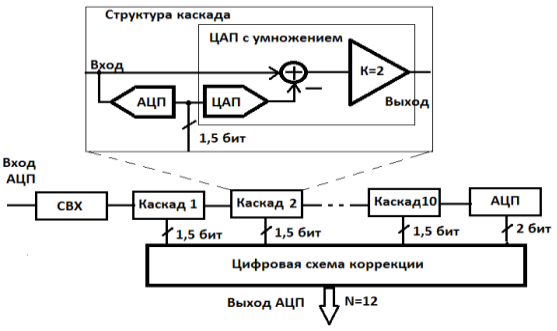


Рис.1.Используемая архитектура в конвейерном АЦП

2.1. Операционный усилитель

Операционный усилитель является одним из наиболее важных компонентов конвейерных АЦП. Коэффициент усиления по постоянному

току и полоса пропускания операционного усилителя определяют достижимую точность и скорость преобразования. Для 12-разрядного конвейерного АЦП коэффициент усиления операционного усилителя с разомкнутым контуром должен быть не менее 70 дБ. Нередко, в практических примерах проектирования можно увидеть коэффициент усиления в 80 дБ. Разработка такого операционного усилителя с высоким коэффициентом усиления при низком напряжении питания является довольно сложной задачей, поскольку традиционное расположение каскадных транзисторов неосуществимо. Использование многокаскадных операционных усилителей с компенсацией приведет к значительному увеличению мощности потребление и снижению скорости. В описываемом АЦП использован полностью дифференциальный двухкаскадный операционный усилитель КМОП, который работает от источника питания 1,8 В. Операционный усилитель обеспечивает коэффициент усиления не менее 80 дБ, частоту единичного усиления не менее 1 ГГц при нагрузке 0,7 пФ и скорость нарастания выходного напряжения до 1000 В/мкс. Потребляемая мощность операционного усилителя составляет не более 8 мВт [4]. Параметры операционного усилителя сведены в таблицу 1.

Таблица 1

Параметр	**Мин. значение**	**Макс. значение**	**Единица измерения**
Напряжение смещения	-232	-35	мкВ
Запас по фазе	37	79	градус
Частота единичного усиления	1,14	3,88	ГГц
Полоса пропускания по уровню -3 дБ	63	160	кГц
Скорость нарастания выходного напряжения (2)	443	1018	В/мкс
Время установления (2)	1,72	5,01	нс
Дифференциальный коэффициент усиления	79,2	94,4	дБ
Коэффициент ослабления синфазного сигнала	72,8	85,9	дБ
Коэффициент подавления нестабильности питания	68,8	100,8	дБ
Ток потребления	2,68	4,19	мА

2.2. ЦАП с умножением

На рисунке 2 показан умножающий цифро-аналоговый преобразователь (MDAC), который является полностью дифференциальной схемой.

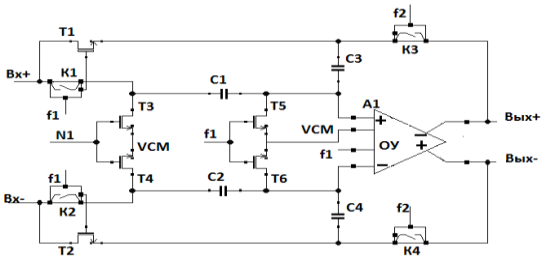


Рис2. Полностью дифференциальный умножающий цифроаналоговый преобразователь

В качестве переключающих элементов использовался ключ типа «bootstrap switch», принципиальная схема которого приведена на рисунке 3.

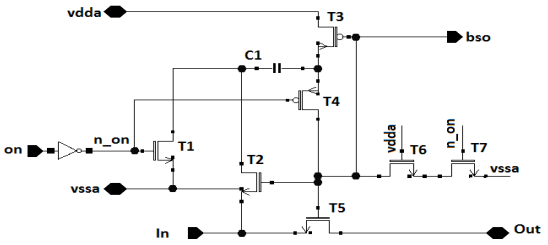


Рис.3. Элемент переключения для MDAC

В момент отпирания коммутирующего транзистора T5 напряжение на его затворе составляет

$$Vg5 = \mathrm{Vs5} + \frac{C1}{C1+Cp} Vdda \qquad (1)$$

где Ср - общая паразитная емкость, подключенная к верхней пластине конденсатора С1.

В качестве последнего каскада в конвейере АЦП используется 2-х разрядный параллельный аналого-цифровой преобразователь.

2.3. Топология АЦП

На рисунке 4 приведена топология СФ-блока КМОП 12-разрядного аналого-цифрового преобразователя конвейерного типа с проектными нормами 28 нм.

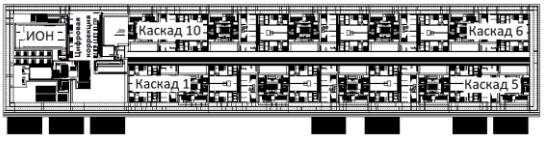


Рис.4. Топология АЦП

Размеры СФ-блока без площадок составляют 630x150 мкм.

3. Результаты моделирования

3.1. Встроенный источник опорного напряжения

Внутренний источник опорного напряжения (ИОН), вырабатывает высокое опорное напряжения VRP = +1,3 В и низкое опорное напряжения VRM = +0,5 В, которые, собственно, и определяют шкалу АЦП. В блоке ИОН имеется возможность тонкой подстройки величины опорного напряжения и его температурной зависимости. Максимальный температурный дрейф в диапазоне от минус 40 С° до плюс 90 С° не превышает 15 ppm/С°.

3.2. Параметры АЦП

Проводилось моделирование АЦП с учетом паразитных элементов конструкции (топологии): резисторов и конденсаторов. Моделирование проводилось в температурном диапазоне от -40 С° до +90 С°, с разбросом питающего напряжения от +1,7 В до +1,9 В и для различных технологических процессов, всего 17 углов. Кроме того, использовался вероятностный метод расчетов Монте Карло.

На рис.5 приведен пример спектра аналогового эквивалента выходного цифрового сигнала АЦП.

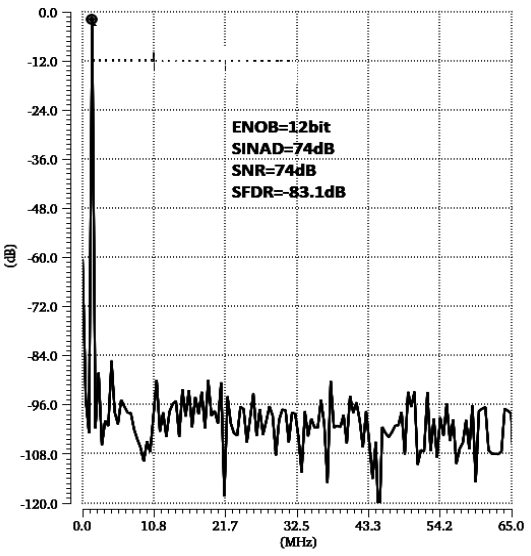
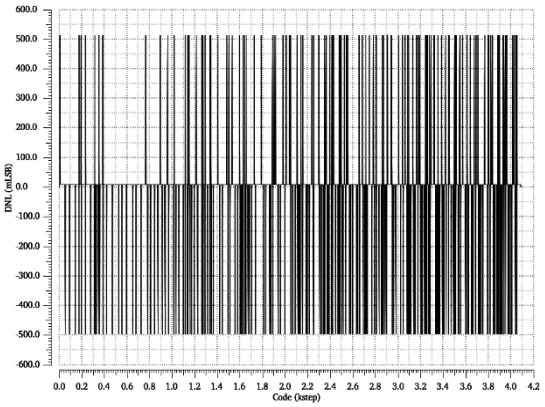
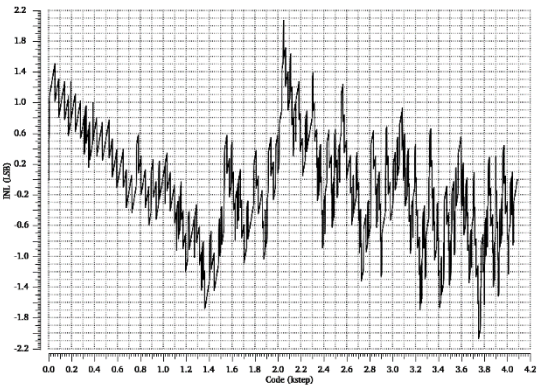


Рис.5. Спектр аналогового эквивалента выходного цифрового сигнала АЦП.

Исследовались статические характеристики АЦП: DNL и INL.



а)



б)

Рис.6. Статические характеристики АЦП:
а -зависимость DNL от выходного кода;
б - зависимость INL от выходного кода.

В таблице 2 приведены параметры АЦП при нормальных условиях.

Таблица.2

Параметр	**Процесс ТТ, Fsampl=125МГц +1,8В; +27С°**		
	Выборка 1	Выборка 2	Выборка 3
ENOB	12,00	11,97	11,98
SINAD, dB	74,00	73.85	73.86
SNR, dB	74,00	73.85	73.86
SFDR, dB	-83,13	-82.24	-82.25
I_{power}, mA	34,24	34,24	34,23

В таблице 3 приведены максимальные отклонения параметров АЦП при всевозможных условиях моделирования в упомянутых выше 17 углах.

Таблица 3

Параметр	**Минимальное значение параметра**	**Максимальное значение параметра**
ENOB	10	12
SINAD, dB	60,4	74,5
SNR, dB	60,4	74,5
SFDR, dB	-85,5	-64,0
I_{power}, mA	24,7	57,4

На рисунке 7 приведено статистическое распределение значение ENOB для 15 выборок в различных углах моделирования.

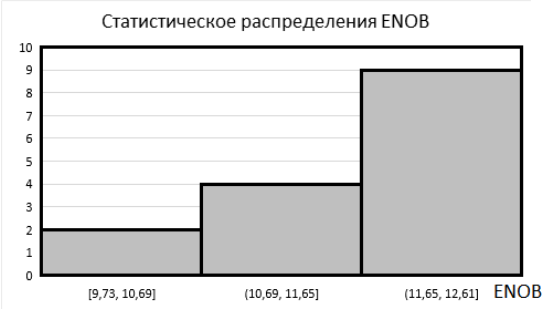


Рис.7. Статистическое распределение значение ENOB для 15 выборок.

3.3. Работа АЦП в режиме субдискретизации

Как упоминалось выше, полоса пропускания АЦП (**FPBW** — *Full Power (Analog) Bandwidth*) является очень важным параметром при построении систем, которые работают в режиме субдискретизации**.** Можно оценить минимально возможное значение полосы пропускания для данного АЦП [5]. За период выборки емкость СВХ должна зарядиться с точностью до одного LSB. Если период выборки равен 1/(2Fs), где Fs – частота выборок, то ошибка выборки сигнала полной шкалы V_{FS} равна:

$$V_{FS} \times \exp\left(-\frac{t}{\tau}\right) = 1 \times LSB$$

Решив относительно t, получаем:

$$t = -\tau \times \ln\left(\frac{1 \times LSB}{V_{FS}}\right)$$

Положив, что τ=1/2πFPBW, определим минимальную полосу АЦП для t=1/(2F_S)

$$FPBW = -\left(\frac{F_S}{\pi}\right) \times \ln\left(\frac{1 \times LSB}{V_{FS}}\right) \quad (2)$$

Для представленного 12 разрядного АЦП с частотой дискретизации 125 Мвыб/с и шкалой 800мВ ограничение снизу для полосы пропускания, рассчитанное по формуле (2) составит FPBW=331 МГц. Следует учитывать, что динамические параметры АЦП как правило ухудшаются с ростом частоты входного сигнала, поэтому оцифровать сигнал из 6-й зоны так же качественно, как из 1-й не получится. На рисунке 8 приведен спектр выходного сигнала при частоте входного сигнала F_{in}=290 МГц. Видно, что в высших зонах Найквиста величина оцифрованного входного сигнала меньше, чем в низших зонах Найквиста.

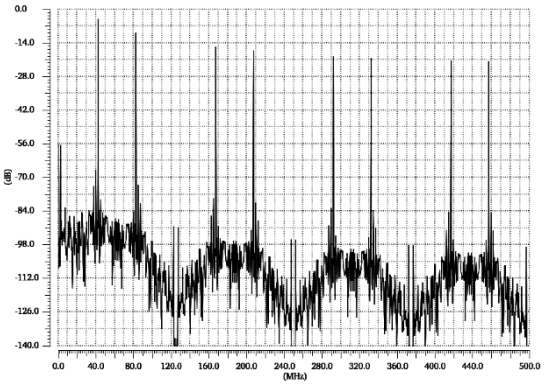


Рис.8. Работа АЦП с частотой входного сигнала вне первой зоны Найквиста.

4. Заключение

В этой статье описывается конструкция конвейерного 12-разрядного АЦП с частотой дискретизации до 125 Мвыб/с, реализованного по КМОП технологии с проектными нормами 28 нм. Показано, что конвейер с переключаемым конденсаторами на 1,5-битных каскадах способен обеспечить линейность 12-разрядного сигнала без искажений или калибровки. Низкое энергопотребление до 100 МВт достигается за счет выбора конфигурации ОУ и оптимизированного по мощности разрешения для первого каскада.

Публикация выполнена в рамках государственного задания НИЦ «Курчатовский институт» — НИИСИ по теме № FNEF-2024-0003 «Методы разработки аппаратно-программных платформ на основе защищенных и устойчивых к сбоям систем на кристалле и сопроцессоров искусственного интеллекта и обработки сигналов».

12-bit analog-to-digital converter of conveyor type

Y.B. Rogatkin

Abstract. A CMOS 12-bit analog-to-digital converter of conveyor type with design standards of 28 nm is presented. Comparatively low power consumption with a sufficiently high sampling frequency of up to 125 MHz is achieved through the use of fully differential two-stage operational amplifiers in the pipeline architecture, operating in a 1.5-bit ADC pipeline architecture per stage with a supply voltage of 1.8V.

Keywords: analog-to-digital converter, sampling rate, bit depth.

Литература

1. Улучшение оцифровки с помощью передискретизации и усреднения, https://microsin.net/programming/dsp/an118-improving-adc-resolution-by-oversampling-and-veraging.html (дата обращения 15.08.2025).

2. Sivaram Prasad Kopparthy,Ishit Makwana, Anu Gupta."Asynchronous 8-bit pipelined ADC for self-triggered sensor based applications", Asia Pacific Conferenceon Postgraduate Research in Microelectronics and Electronics, 2012.

3. S. Lewis et al., “A 10-b 20 MSamples/s analog to digital converter,”IEEE J. Solid-State Circuits, vol. 27, pp. 351–358, Mar. 1992.

4. Ю.Б.Рогаткин. Двухкаскадный операционный усилитель для аналого-цифрового преобразователя конвейерного типа. «Нано- и микросистемная техника», 2025, №4, с.201-208.

5. https://habr.com/ru/companies/milandr/articles/528164 (дата обращения 15.08.2025).

Теплопроводность кремниевого полевого GAA нанотранзистора с учетом шероховатости границы

Н.В. Масальский

НИЦ «Курчатовский институт» — НИИСИ, Москва, Российская Федерация;

volkov@niisi.ras

Аннотация. Обсуждается тепловая модель для кремниевых полевых GAA нанотранзисторов с учетом тепловых воздействий, вызванных шероховатостью границы. Модель построена на основе метода теории возмущений, в котором учитывается влияние зависимости диаметра нанопроволоки и шероховатости поверхности на теплопроводность канала транзистора, а также влияние особенностей GAA нанотранзисторной структуры на рассеивание тепла. В данном случае для тепловой коррекции модели транзистора (идеализированной), для которой разработана апробированная математическая модель, следует добавить слабое возмущение через дополнительный "возмущающий" гамильтониан. Тогда различные физические величины, связанные с возмущенной системой, могут быть выражены как "поправки" к характеристикам исходной модели. Эти поправки в априори малы по сравнению с размером самих величин. Однако они 1) существенно изменяют характеристики исходной системы, 2) упрощают алгоритмы их вычисления. На основе модели обсуждается влияние параметров конструкции GAA нанотранзисторов на его тепловыделение. Разработанная модель может быть применена для проектирования схем на основе GAA нанотранзисторов с учетом тепловых факторов.

Ключевые слова: кремниевый gate-all-around (GAA) нанотранзистор, теплопроводность, шероховатость границы, корреляционная длина, среднеквадратичное отклонение

1. Введение

Современные транзисторные технологии позволяют создавать уникальные полупроводниковые структуры, которые отличаются превосходными электрофизическими характеристиками [1-3]. Такой качественный скачок стал возможен в результате перехода от планарных к 3-х мерным архитектурам Одной из потенциально значимых архитектур является кремниевый нанопроволочный полевой транзистор с полностью охватывающим затвором (gate-all-around (GAA). Однако его сильный нагрев может приводить к снижению производительности и к возникновению проблем с надежностью, которые будут проявляться из-за ограничений, связанных с геометрией устройства и повышенного рассеяния фононов на границах транзистора.

Природа этого рассеяния связана с конкурентностью фундаментальных параметров «средним свободным пробегом» теплоносителей и соответствующим характерным размером нанотранзистора [4]. Поэтому теплопроводность нанопроволочного GAA транзистора будет зависеть от формы и размера рабочей области (РО) [5, 6], которая представляет собой в данном случае цилиндрическую нанопроволку (см. рис. 1). Теплопроводность кремниевых нанопроволок определяется критическими параметрами транзисторной структуры, такие как длина и диаметр РО, шероховатость ее поверхности [7-9]. В этой статье рассмотрены особенности конструкции GAA нанотранзистоной структуры и зависящая от нее теплопроводность РО. Основываясь на этом подходе, можно исследовать доминирующие факторы, влияющие на термоэффекты, которые важны для проектирования устройств и схем с учетом температурных условий [10, 11].

Цель настоящей работы - с помощью теории возмущения численно исследовать аномальную теплопроводность кремниевых цилиндрических наноструктур с учетом шероховатости границы в стационарных состояниях. При этом тепловые характеристики материала учитываются посредством его объемной теплопроводности. Следует отметить, экспериментально установлена следующая оценка: теплопроводность кремниевых нанопроволок более чем на 2 порядка ниже объемного значения вследствие эффектов масштабирования и высокого отношения поверхности к объему [9, 12-14].

Точное значение коэффициента теплопроводности можно рассчитать, используя комплексный подход для моделирования рассеяния на поверхности фононов в тонких нанопроволках, который основан исследовании случайного сценария распределения шероховатостей на поверхности РО в совокупности с геометрическим представлением шероховатости границы РО для рассеяния фононов, которое напрямую связано с параметрическим описанием шероховатости. Вли-

яние шероховатости ощутимо проявляются тогда, когда характерные размеры неровностей соизмеримы с длиной волны фонона или электрона. В нашем понимании изменение термосвойств нанопроволки связано с тем, что скорость рассеяния фононов на поверхности изменяется в зависимости от технологических параметров. Такой подход имеет решающее значение для исследования переноса фононов в тонких слоях GAA транзисторов, где можно использовать точные модели, основанные на теории возмущений [8, 15, 16], которые напрямую связаны с физическими свойствами поверхности раздела PO (кремний) -охватывающий оксид (оксид кремния). Достоверность вероятностного распределения шероховатости на поверхности раздела может быть экспериментально подтверждена с помощью просвечивающей электронной микроскопии.

2. Моделирование теплопроводности PO

Чтобы точно смоделировать температурные эффекты, необходимо учитывать рассеяние фононов на границе и рассеяние фононов на шероховатости поверхности. В тепловом моделировании эти механизмы должны быть скрупулёзно отражены с учетом особенностей структуры кремниевого цилиндрического GAA нанотранзистора. Их совместное с геометрией PO влияние будет изменять тепловые потоки, что в конечном итоге скажется на теплопроводности PO и найдет свое отражение в поведении коэффициента теплопроводности. Хотя нет достоверных данных о возможных типах шероховатости нанопроволок, опираясь на различные теоретические и экспериментальные исследования, мы для диапазона длин PO в несколько десятков нанометров установили диапазон среднеквадратичные значения шероховатости Δ от 0.5 нм до 2 нм. И выбрали две длины корреляции Λ, которая связана со средним расстоянием между пиками неровности на границе раздела Si-SiO2, 5 и 10 нм, что в целом согласуется с данными электронной микроскопии. На практике эти параметры, зависящие от техпроцесса, определяются экспериментально. Диаметр PO варьируется от 10 до 30 нм, что соответствует прототипам для современных нанотранзисторных СБИС. Длина PO выбирается из условия подавления коротко-канальных эффектов [2]. В настоящем исследовании фиксированными параметрами для всех случаев являются концентрация легирования PO 1x1015 см-3 (низколегированный случай), толщина охватывающего диэлектрика (оксида кремния) 5 нм и окружающая температура T 300 K.

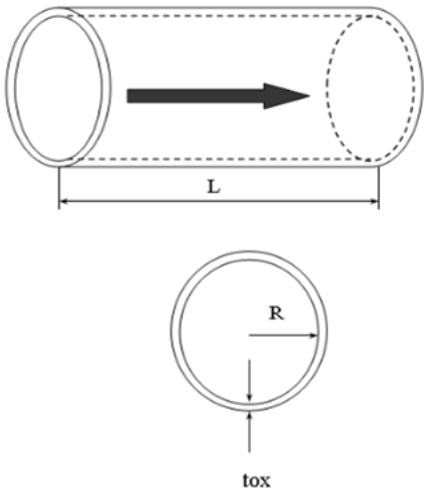


Рис. 1. Эскиз цилиндрической PO с круглым поперечным сечением и с внутренней кремниевой областью полностью охваченной пленкой оксида кремния, где L – длина, R – радиус, tox – толщина оксида кремния. Стрелкой показано направление теплового потока.

3. Модель переноса фононов

В общем случае тонкой нанопроволоки изменение ее поперечного размера (в перпендикулярном направлении) влияет на перенос фононов. Математически это эквивалентно возмущению гамильтониана системы. В предположении, что 1) пограничное рассеяние является в основном упругим процессом при достаточно длительном времени t по сравнению со временем релаксации энергии и 2) никакие фононы не испускаются в окружающую среду, т.е. нанопроволка охвачена пленкой с более низкой теплопроводностью, что характерно в транзисторных структурах, где кремниевая PO транзистора (теплопроводность которой 149 Вт/(м·К)) окружена пленкой оксида кремния с теплопроводностью 1,4 Вт/(м·К) – это в 100 раз ниже кремния !!! Мы рассматриваем рассеяния фононов, обусловленное механизмом шероховатости границы раздела Si-SiO2 (ШГР) [5, 6, 8], интерпретируя этот механизм в виде серию раздельных искривлений поверхности, которые воспринимаются фононами в виде центров рассеяния вдоль направления своего распространения. Эти акты рассеяния описываются при помощи теории возмущения, где, используя возмущенный гамильтониан, рассчитываются вероятности изменения импульса k и частоты ω в единицу времени падающего фонона в новое состояние с импульсом k' и частотой ω' [17]. Наличие пространственной шероховатости поверхности приводит к изменению частоты в ортогональной плоскости так:

$$\omega(k) = \omega 0(k)(1 - \gamma\Delta) \quad (1)$$

где γ – подгоночный параметр с размерностью нм$^{-1}$, ω0(k) - дисперсия фононов невозмущенного гамильтониана.

В [18] показано, что автокорреляционная функция шероховатости поверхности кремния Si представляется функцией Гаусса, которая, согласно известной теореме Винера-Хинчина, приблизительно описывается выражением вида:

$$\Delta(q) = \pi\Delta^2\Lambda^2 \exp(-\frac{1}{4}(q\Lambda)^2), \qquad (2)$$

где del(q) является преобразованием Фурье пространственного возмущения, равным

$$\Delta(q) = \int d\mathbf{r}\Delta(\mathbf{r})\exp(i\mathbf{qr})$$

Зависимость возмущенного гамильтониана H’ от частоты имеет вид H’~ ω'2.

В обсуждаемом подходе рассматриваются переходы между всеми акустическими и оптическими ветвями. Дополнительное тепловое изменение скорости рассеяния фононов на поверхности описывается статистикой Бозе-Эйнштейна [18]. В данном случае число заполнения включая температурную зависимость процесса рассеяния определяется выражением

$$\langle n \rangle = 1 \Big/ (\exp(\hbar\omega \Big/ k_B T - 1)),$$

где h – постоянная Планка, kB- константа Больцмана.

В данном случае температура ограничивает заполнение каждого частотного диапазона, тем самым восстанавливая эффект заполнения ветвей с более низкой энергией при низкой температуре. Поэтому теплопроводность различных поперечных сечений будет совокупностью вкладов отдельной ветви [19]. С учётом того, спектральное выражение (2) симметрично и преобладающими являются рассеяния зеркального типа вклад в теплопроводность ветви i:

$$k_i(T) = \frac{1}{3}\int_i dE \frac{d\langle n \rangle}{dT} EN_i(E)Vs_i(E)\times \qquad (3)$$
$$\times v_i^{tot}(E)$$

где Ni(E) - фононная плотность состояний в i-й ветви, энергия E'(k') проходит вдоль j -й ветви. Общая скорость рассеяния vi(E), начиная с ветви i, равна сумме значений vi,j(E) по всем ветвям j, Vsi - скорость звука, подчеркнем – она зависит от направления распространения. Отметим, что скорость рассеяния на границе зависит от скорости звука в данной ветви и, следовательно, зависит от частоты. При этом существует диапазон частот, в котором фононы вносят больший вклад в рассеяние на шероховатости поверхности. Скорость рассеяния фононов от ветви i к ветви j определяется выражением, где объемный интеграл по k'-пространству сводится к поверхностному интегралу [18].

Чтобы воспроизвести измеренное физическое поведение нанопроволки в исследуемых диапазонах, при расчете времени рассеяния были рассмотрены механизмы рассеяния по методу Умклаппа [20], который показал хорошее соответствие с экспериментами. Допущения: 1) нормальное рассеяние в продольных акустических ветвях учитывается соотношением $8\times10^{-45}\omega^4$, которое соответствует экспериментальным данным с большей точностью и 2) для скорости звука в акустических ветвях используется аналитическое выражение из [21] благодаря его хорошему согласованию с данными объемного кремния.

Для полного учета частотной зависимости возмущенного гамильтониана необходимо учитывать всеобъемлющую дисперсию фононов [22, 23]. Метод из [23] применяется для вычисления фононной плотности состояний на основе дисперсионного соотношения, упомянутого выше. Для вычисления поверхностных интегралов первая зона Бриллюэна аппроксимируется решеткой размером 30×30×30, что обеспечивает приемлемую точность [23].

Скорости рассеяния фононов вычисляются для фиксированных значений параметров D и Λ для различных значений параметра Δ из выбранного диапазона. На рис. 2 представлены расчетные зависимости времени релаксации фононов τ от Δ для D=30 нм и Lg=80 нм.

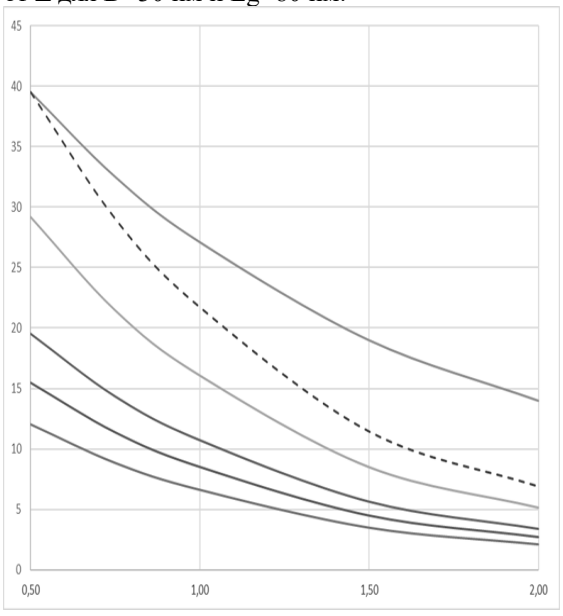


Рис. 2. Зависимость время релаксации фононов (пс) от Δ (нм) при разных энергиях активации (сверху вниз) 10, 15, 20, 25, 30 мэВ (сплошные линии), пунктир – времени релаксации при одинаковом равномерном темпе заполнения ветвей.

В приведенных выше условиях среднее время релаксации фононов, обусловленное только механизмами ШГР, для разных уровней энергий колеблется от 30 пс до 5 пс. Следует отметить, что характерной зависимость τ(E) аппроксимируется полиноминальной зависимостью приведенной на рис. 3. Согласно которой с ростом энергии E время релаксации квадратично уменьшается.

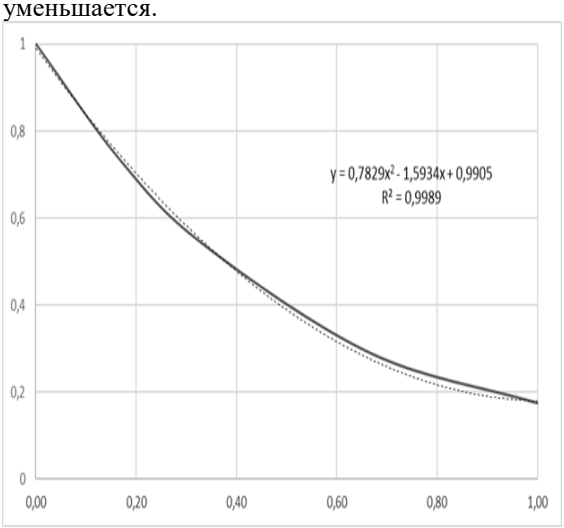


Рис. 3. Время релаксации фононов (нормир.) от энергии активации (нормир.) при Δ = 1 нм

В дальнейших расчетах мы используем уже аппроксимирующее выражение. При этом на начальной стадии при расчете начального приближения применяется более упрощенное выражение вида $\tau_o = aE_o^2 - 2aE_o + 1$, где τ_o, E_o - нормированные значения. Из (2) следует, с увеличением параметра Λ при условии Δ/Λ <<1 благоприятствует процессам переноса фононов и время жизни возрастает. Наоборот, в пределе сильной шероховатости при Δ/Λ>1 гауссово распределение спектральной плотности не будет отражать реальный процесс рассеяния. Здесь вклад квадратичного члена Λ^2 будет усреднять влияние корреляционной длины. Поэтому вместо выражения (2) необходимо использовать другое. Однако данный случай лежит далеко от рассматриваемых диапазонов, и мы не будем его анализировать в рамках данной работы.

Отметим, что в [22, 23] указано, что при низких температурах (меньше 100 К) точность вычислений будет снижается из-за возрастания количества итераций. В настоящей работе этот результат тоже нашел подтверждение, при этом шаг сетки при T=100 К уменьшается в 2.5 раза по сравнению с первоначальным случаем при T=300 К.

4. Коэффициент теплопроводности

Численное моделирование было сфокусировано на выявлении зависимости коэффициента теплопроводности РО нанотранзистора от ее диаметра и параметра при двух длинах корреляции 5 и 10 нм. Такой выбор значений позволяет охватить большинство практически значимых технологий изготовления кремниевых нанопроволок. Численное моделирование выполнено для 200 прототипов.

На рис. 4 и 5 приведены извлеченные из результатов моделирования зависимость коэффициента теплопроводности РО от ее диаметра при фиксированных значениях Δ и Λ = 5 нм. В данном случае теплопроводность каналов кремниевых нанопроволок уменьшается с увеличением диаметра. При этом крутизна зависимости kw(D) возрастает с со снижением величины параметра Δ. Следовательно, когда шероховатость поверхности РО уменьшается, теплопроводность нанопроволоки существенно возрастает в диапазоне диаметров в несколько десятков нанометров и технологически обоснованных длин корреляции.

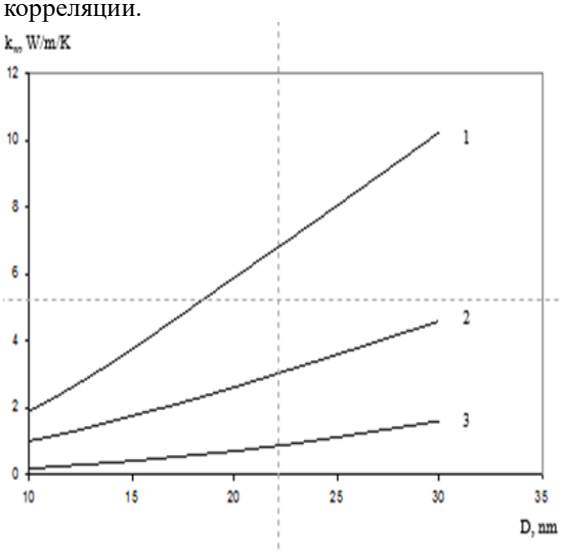


Рис. 4. Зависимость коэффициента теплопроводности kw от диаметра D при фиксированной длине корреляции 5 нм и разных шероховатостях 1- 0.5 нм, 2 – 1 нм, 3 – 2 нм

Следует отметить, что с уменьшением диаметра возрастает чувствительность к параметрам шероховатости поверхности в большей степени к Λ и меньшей к Δ. Кроме того, квази-линеаризованный подход остается в силе до тех пор, пока возмущения остаются малыми по сравнению с полной энергией фононов.

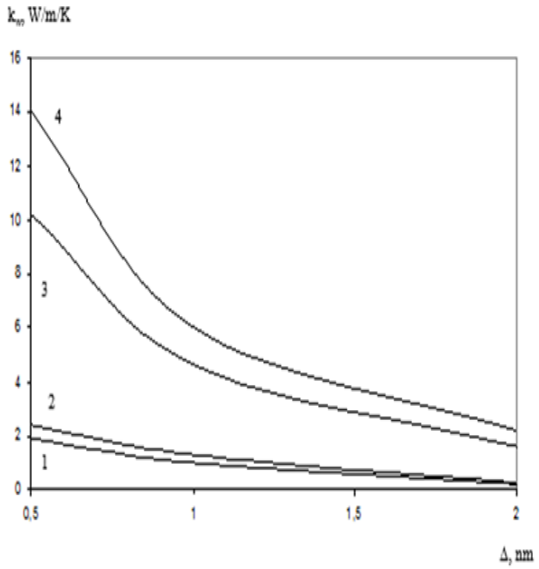


Рис 5. Зависимость коэффициента теплопроводности kw (Δ) для разных диаметров и длин корреляции

1- 10 нм и 5 нм, 2 – 10 нм и 10 нм, 3 – 30 нм и 5 нм, 4 – 30 нм и 10 нм.

Из результатов моделирования следует, что подход, основанный на теории возмущений, устанавливает зависимость kw от (Δ/D)-2. При этом в области больших диаметров наблюдается типичное линейное масштабирование вида (D/Δ). В области малых диаметров снижается теплопроводность при одновременном влиянии значения диаметра и шероховатости поверхности. Общий вклад шероховатости поверхности в ограничение теплопроводности увеличивается в несколько раз при снижении диаметра при фиксированных Λ и Δ. Увеличение длины корреляции приводит к росту коэффициента теплопроводности и к изменению формы зависимости kw (Δ). Ее крутизна заметно возрастает. В литературе обсуждается проблема определения так называемого критического диаметра, ниже которого зависимость теплопроводности кремниевой нанопроволки, полностью охваченной теплоизолирующем слоем, отклоняется от классического линейного приближения. В рассматриваемых диапазонах D, Λ и Δ его определение неактуально, поскольку эта граница всегда расположена возле наибольших их значений. Подчеркнем, что наши выкладки справедливы при условии D>> Δ диапазона D от 10 нм и выше.

Рассмотренный подход позволяет предсказывать температурную зависимость теплопроводности, которая зависит от скорости рассеяния на шероховатости при окружающих температурах отличных от нормальной. На рис. 6 представлена расчетная зависимость аналогичная зависимости 2 на рис. 4, но при T=200 K.

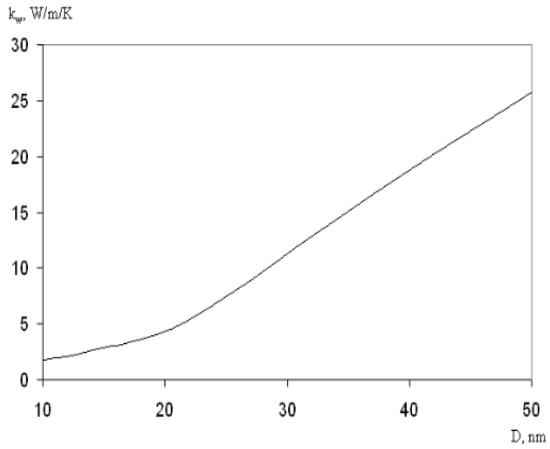


Рис. 6. Зависимость коэффициента теплопроводности kw(D) при Λ = 5 нм и Δ = 1 нм и T=200 K

Из которой предсказывается существенное повышение коэффициента теплопроводности. Таким образом, при низких температурах низкочастотные фононы практически не участвуют в процессе рассеяния на границе раздела «РО-оксид кремния»

Дополнительно можно сделать вывод о том, что диаметр РО и длина области сток/исток GAA нанотранзисторов с цилиндрической геометрией являются доминирующими факторами, влияющими на рассеивание тепла устройством. Толщина поликремниевого затвора и толщина области сток/исток будут незначительно влияют на термоэффекты. Отвод тепла из канала в область затвора блокируется. В этом случае тепло в РО практически целиком передается через РО в область сток/исток. Следует отметить, что свойства теплопередачи кремниевой РО важно для исследования механизма самонагревания GAA нанотранзисторов и его теплового паразитного электрического сопротивления.

5. Заключение

Проанализировано сильное влияние шероховатости поверхности на теплопроводность тонких кремниевых цилиндрических GAA нанотранзисторов. На основе соотношения полной дисперсии фононов исследована частотно-зависимая модель рассеяния фононов на границе раздела «кремниевая рабочая область - оксид кремния». Полученные в результате моделирования данные о теплопроводности предсказывают, что при малых диаметрах наблюдается значительное отклонение теплопроводности от линейной зависимости от диаметра до величины (Δ/D)-2 из-за влияния параметрической шероховатости. При этом диаметр рабочей области и длина области сток/исток GAA нанотранзисто-

ров с цилиндрической геометрией являются доминирующими факторами, влияющими на рассеивание тепла устройством. Кроме того, представляется важным, что эффект шероховатости поверхностей должен быть сильнее в тонких слоях и изменяться в зависимости от частоты падающих фононов.

Публикация выполнена в рамках государственного задания НИЦ «Курчатовский институт» — НИИСИ по теме № FNEF-2024-0003 «Методы разработки аппаратно-программных платформ на основе защищенных и устойчивых к сбоям систем на кристалле и сопроцессоров искусственного интеллекта и обработки сигналов».

Thermal Conductivity of a Silicon GAA Field-Effect Nanotransistor, Taking into Account the Roughness of the Boundary

N.V. Masalsky

Abstract. A thermal model for silicon field-effect GAA nanotransistors is discussed, taking into account the thermal effects caused by boundary roughness. The model is based on the perturbation theory method, which takes into account the effect of the dependence of the nanowire diameter and surface roughness on the thermal conductivity of the transistor channel, as well as the effect of the characteristics of the GAA nanotransistor structure on heat dissipation. In this case, for thermal correction of the (idealized) transistor model, for which a proven mathematical model has been developed, a weak perturbation should be added through an additional "perturbing" Hamiltonian. Then the various physical quantities associated with the perturbed system can be expressed as "corrections" to the characteristics of the original model. These corrections are a priori small compared to the size of the quantities themselves. However, they 1) significantly change the characteristics of the initial system, and 2) simplify the algorithms for their calculation. Based on the model, the influence of the nanotransistor chip design parameters on its heat dissipation is discussed. The developed model can be used to design circuits based on nanotransistor chips, taking into account thermal factors.

Keywords: silicon gate-all-around (GAA) nanotransistor, thermal conductivity, boundary roughness, correlation length, mean square deviation

Литература

1. Nanoelectronics: Devices, Circuits and Systems. Editor by Brajesh Kumar Kaushik. Elsevier, 2018.
2. G. Tomar, A. Barwari. Fundamental of electronic devices and circuits. Springer, 2019.
3. International Technology Roadmap for Semiconductors (ITRS) Interconnect, 2020 Edition. [Online] Available: https://irds.ieee.org/editions/2020 (accessed on 25 September 2023).
4. J.H. Davies. The physics of low dimensional semiconductors. New York: Plenum, 1998.
5. D. Li, Y. Wu, P. Kim, L. Shi, P. Yang, A. Majumdar. Thermal conductivity of individual silicon nanowires // “Appl. Phys. Lett.”, (2003), V. 83, No.14, 2934-2936.
6. X. Yang, A.C To, R. Tian. Anomalous heat conduction behavior in thin finite-size silicon Nanowires // “Nanotechnology”, (2010), V. 21, No. 15, 155704.
7. P.K. Schelling, SR. Phillpot, P. Keblinski. Comparison of atomic-level simulation methods for computing thermal conductivity // “Phys. Rev. B”, (2002), V. 65, No. 14, 144306.
8. P. Martin, Z. Aksamija, E. Pop, U. Ravaioli. Impact of phonon-surface roughness scattering on thermal conductivity of thin Si nanowires // “Phys. Rev. Lett.”, (2009), V. 102, No.12, 125503.
9. L. Liu, X. Chen. Effect of surface roughness on thermal conductivity of silicon nanowires // “J. Appl. Phys.”, (2010), V. 107, No. 3, 033501.
10. A. Vassighi, M. Sachdev. Thermal and power management of integral circuits. New York, NY, USA: Springer-Verlag, 2006.
11. D. Wolpert, P. Ampadu. Managing temperature effects in nanoscale adaptive system. New York: Springer-Verlag, 2012.
12. A.I. Boukai, Y. Bunimovich, J. Tahir-Kheli, J.-K. Yu, W.A. Goddard III, J.R. Heath. Silicon nanowires as efficient thermoelectric materials // “Nature”, (2008), V. 451, 168-171.
13. A.I. Hochbaum, R. Chen, R.D. Delgado, W. Liang, E.C. Garnett, M. Najarian, A. Majumdar, P. Yang. Enhanced thermoelectric performance of rough silicon nanowires // “Nature”, (2008), V. 451, 163-167.

14. R. Wang, J. Zhuge, R. Huang, T. Yu, J. Zou, D.-W. Kim, D. Park, Y. Wang. Investigation on variability in metal-gate Si nanowire MOSFETs: Analysis of variation sources and experimental characterization // “IEEE Trans. Electron Devices”, (2011), V. 58, No. 8, 2317-2325.

15. Z. Aksamija, U. Ravaioli. Energy Conservation in Collisional Broadening, Simulation of Semiconductor Processes Devices. Vienna: Springer, 2007.

16. Thermal Nanosystems and Nanomaterials. Topics in Applied Physics. Editor by S. Volz. Berlin: Springer, 2010.

17. J. Ziman. Electrons and phonons: the theory of transport phenomena in solids. Clarendon,
1960.

18. C. Kittel. Introduction to Solid State Physics. New York: Wiley, 2005.

19. F.X. Alvarez, D. Jou, A. Sellitto A. Phonon boundary effects and thermal conductivity of rough concentric nanowires // “J. Heat Transf. T. ASME”, (2011), V. 133, 022402.

20. N. Mingo. Calculation of Si nanowire thermal conductivity using complete phonon dispersion relations // “Phys. Rev. B”, (2003), V. 68, 113308.

21. E. Pop, R. Dutton, K. E. Goodson. Analytic band Monte Carlo Model for electron transport in Si including acoustic and optical phonon dispersion // “Journal of Applied Physics”, (2004), V. 96, No. 9, 4998-5005.

22. M. Asheghi, Y.K. Leung, S.S. Wong, K.E. Goodson. Phonon-boundary scattering in thin silicon layers // “Appl. Phys. Lett.”, (1997), V. 71, 1798–1800.

23. Z. Aksamija, U. Ravaioli. Joule heating and phonon transport in nanoscale silicon MOSFETs // “IEEE International Conference on Electro/Information Technology”, (2007), V. 1, 70-72.

17. J. Ziman, Electrons and phonons: the theory of transport phenomena in solids (Clarendon, 1960), 1st ed.

18. C. Kittel, Introduction to Solid State Physics, New York: Wiley (2005).

19. C. Glassbrener and A. Slack, Phys. Rev. 134, 1058 (1964). Alvarez, F.X., Jou, D., Sellitto, A.: Phonon boundary effects and thermal conductivity of rough concentric nanowires. J. Heat Transf. T. ASME 133, 022402 (7 pp.) (2011)

20. N. Mingo, Phys. Rev. B 68, 168 (2003). Mingo, N. Calculation of Si nanowire thermal conductivity using complete phonon dispersion relations. Phys. Rev. B 68, 113308

21. E. Pop, R. Dutton, and K. Goodson, Journ. App. Phys. 96, 4998 (2004). E. Pop, R. Dutton and K. E. Goodson, “Analytic Band Monte Carlo Model for Electron Transport in Si including Acoustic and Optical Phonon Dispersion,” Journal of Applied Physics, Vol. 96, No. 9, 2004, pp. 4998-5005.

22. W. Weber, Phys. Rev. B 15 (1970). Asheghi, M., Leung, Y.K., Wong, S.S., Goodson, K.E.: Phonon-boundary scattering in thin silicon layers. Appl. Phys. Lett. 71, 1798–1800 (1997)

23. Z. Aksamija, U. Ravaioli, Joule Heating and Phonon Transport in Nanoscale Silicon MOSFETs. IEEE International Conference on Electro/Information Technology. 2007. 1. P.70-72.

Ускорение операции быстрой свёртки для множества комплексных векторов на основе технологии OpenCL

А.А. Бурцев

НИЦ «Курчатовский институт» — НИИСИ, Москва, Российская Федерация;

burtsev@niisi.msk.ru

Аннотация. Статья посвящена применению технологии OpenCL, позволяющей использовать мощные ресурсы графических процессоров для повышения быстродействия вычислительных программ. Рассматриваются приёмы разработки в среде OpenCL эффективных параллельных программ, позволяющих ускорить операцию свёртки для множества комплексных векторов на основе быстрого преобразования Фурье.

Ключевые слова: параллельное программирование, технология OpenCL, гетерогенные системы, быстрое преобразование Фурье, операция свёртки.

1. Введение

На современных компьютерных установках для разработки параллельных программ обычно применяются широко известные технологии, ориентированные на исполнение таких программ в многопроцессорной среде (OpenMP [1, п. 5.1]) или в среде из нескольких связанных сетью компьютеров (MPI [1, п. 5.2]). Но наряду с ними предлагаются также и такие технологии, которые позволяют использовать мощные ресурсы графических процессоров для повышения быстродействия вычислительных программ. Так что сегодня можно ускорять вычисления даже на одном компьютере, если в его составе предусмотрена видеокарта, поддерживающая одну из этих технологий. Например, технологию OpenCL [2].

Знакомство с технологией OpenCL было предложено автором в серии ранее опубликованных им статей [3-7]. В них демонстрировались приёмы разработки эффективных параллельных программ для решения ряда вычислительных задач. В частности, рассматривались варианты построения по технологии OpenCL программ, ускоряющих перемножение больших матриц [3], вычисление интегралов [4], а также программ, ускоряющих в среде OpenCL выполнение операции быстрого преобразования Фурье (БПФ) как для одиночных комплексных векторов [5-6], так и для их множества [7].

Причём для множества векторов получалось добиться ускорения операции БПФ даже в большей степени, чем для одиночных векторов, если удавалось подключить к обработке вычислительной программы в среде OpenCL как можно больше параллельных исполнителей.

Решение многих задач цифровой обработки сигналов строится на основе применения операции быстрого преобразования Фурье. И если уж с помощью технологии OpenCL можно значительно ускорить выполнение самой операции БПФ, то следующим встаёт вопрос, а в какой степени удастся ускорить те вычислительные программы, которые используют БПФ в качестве базовой операции?

Проведём исследование этого вопроса на примере разработки одной из таких программ. Рассмотрим программу, выполняющую операцию быстрой свёртки, выполняемой над множеством комплексных векторов. В такой программе для выполнения свёртки каждой пары комплексных векторов требуется (помимо других операций) по три раза вызывать операцию БПФ для векторов той же длины.

Как и прежде, представим сначала алгоритмы, по которым осуществляется такая множественная операция быстрой свёртки в обычных программах, рассчитанных на последовательное их исполнение одним процессором. А затем рассмотрим разнообразные варианты программ, позволяющих ускорить исполнение такой операции множественной свёртки в среде OpenCL. А чтобы проанализировать показатели производительности OpenCL-программ разных вариантов, прогоним варианты этих программ на различных OpenCL-платформах для одних и тех же наборов данных, построим для наглядности таблицы полученных коэффициентов ускорения, и затем, сравнив их результаты, выясним, какой из вариантов OpenCL-программы обеспечивает наилучшие показатели ускорения на той или иной OpenCL-платформе.

2. Базовая программа быстрой свёртки комплексных векторов многомерного массива

Операция свёртки двух комплексных векторов X[L] длины L и Y[S] длины S заключается в получении третьего вектора Z[N] длиной N=L+S-1, элементы Z_K (k=0,..,N-1) которого вычисляются по формуле [8,с.147]:

$$Z_k = \sum_{i=0}^{S-1} \{Y_i \times X_{k-i}\}$$

Операция свёртки (convolution) над векторами (последовательностями чисел) обозначается сокращённо символом * Z[N]=X[L]*Y[S].

Сложность вычисления свёртки по этой формуле оценивается порядком O(N·S). Можно, однако, заметно ускорить вычисление свёртки путём применения быстрого преобразования Фурье (БПФ). Такой вариант вычисления получил название быстрая свёртка [8,с.355]. Поясним подробнее, как она осуществляется.

Заданные векторы X[L] и Y[S] дополним нулями до длины ближайшей степени двойки ($N=2^P$, N≥L+S-1). Применим к каждому вектору X[N] и Y[N] операцию БПФ (*FFT*):

X'[N]= *FFT*(X[N]); Y'[N]= *FFT*(Y[N]);

Получившиеся векторы X'[N] и Y'[N] перемножим покомпонентно, получив вектор Z'[N]:

$Z'_K = X'_K \times Y'_K$ для всех k=0,..,N-1.

А теперь применим к вектору Z' обратное БПФ (*iFFT*) и получим вектор Z[N]:

Z[N]= *iFFT*(Z'[N]),

который и будет результатом свёртки.

Сложность вычисления быстрой свёртки по такому алгоритму оценивается порядком $O(N \cdot \log_2 N)$, как и сама операция БПФ.

С алгоритмом прямого БПФ мы уже не раз знакомились ранее (например, см. описание функции FFT в [5, п.2.2]). Он основан на многократном исполнении так называемых операций «бабочек Фурье» BF(A,B,V) над всевозможными парами комплексных чисел A и B с весовым коэффициентом V вида:

$$W_N^u = e^{-i \cdot 2\pi \cdot u/N}$$

Для обратного БПФ нужно исполнить такой же алгоритм БПФ, но в качестве весовых коэффициентов взять сопряженные им комплексные числа (т.е. с другим знаком в мнимой части):

$$W_N^u = e^{+i \cdot 2\pi \cdot u/N}$$

а в завершении провести так называемую нормализацию, поделив значения полученных элементов вектора на величину его длины N.

Предполагается, что значения весовых коэффициентов, используемые алгоритмом БПФ, вычисляются заранее и приготавливаются в виде таблицы – массива комплексных чисел W (размером от 0 до N). Причём значение, сопряжённое значению W[u], можно взять из этой таблицы на такой же позиции с конца: W[N-u].

Учитывая это, подкорректируем алгоритм функции FFT. Добавим в заголовок функции параметр D, задающий направление БПФ: D=1 для прямого и D=0 для обратного преобразования Фурье. И будем выбирать коэффициент V из таблицы W, проверяя условие: если D=1, то V=W[u], иначе V=W[N-u]. А при завершении всего преобразования добавим выполнение нормализации, если D=0. После такой модификации функцию FFT можно будет применять для выполнения не только прямого (с D=1), но и обратного (с D=0) БПФ.

```
void FFT(int N,complex *X,
 complex *Y,complex *W, int D){
complex *VX,*VY,V,T;
int P= iLog2(N); float Norm;
//1. Бит-реверсная перестановка:
 BRVPRST(N,P,X,Y);
//2.Основной цикл исполнения бабочек Фурье:
 int t,s,h,G,R,d,k,u,a,b;
 s=1;h=2;G=1;R=N/2;d=N/2;
 for (t=1; t<=P; t++) {
 // s=2^(t-1); h=2^t; G=2^(t-1); R=2^(P-t)
 for(k=0,u=0;k<G;k++,u=u+d){
 if(D) V=W[u];else V=W[N-u];
 for(i=0,a=k; i<R; i++,a=a+h)
 {b=a+s; BF(Y[a],Y[b],V);}
 }//for k
 h=h*2;s=s*2;G=G*2;R=R/2;d=d/2;
 }//for t
// 3. Нормализация (для обратного БПФ)
 if(!D){ Norm=1.00/(float)N;
 for (i=0;i<N;i++)
 Y[i]= cMScal(Y[i],Norm);
 }//if !Dir
}//FFT
```

В теле этой функции используется вызов макроопределения **BF** для исполнения операции «бабочки Фурье» и функция **cMScal** для умножения комплексного числа на вещественный скаляр. Приведём здесь их возможные реализации (для версии языка Си, имеющей встроенный тип complex):

```
#define BF(A,B,V) { complex T; \
 T=B*V; B=A-T; A=A+T; }
complex cMScal(complex C, float S)
{ complex R; R.Re= C.Re*S;
 R.Im= C.Im*S; return R; }
```

Для бит-реверсного копирования комплексного вектора применяется вспомогательная процедура **BRVPRST**:

```
void BRVPRST(int N, int L,
  complex *X,complex *Y) {
 unsigned int i,j;
 for (i=0; i<N; i++) {
 BRvrsVal(j,i,L); Y[j]=X[i];
 }// for i
}//BRVPRST
```

Макрос **BRvrsVal**(j,i,L) здесь используется для получения бит-реверсного значения j длины L (т.е. записанного в обратном порядке значения в битовом коде длины L) для заданного индекса i. Приведём один из возможных вариантов определения такого макроса:

```
#define BRvrsVal(k,i,L){ k=i;\
k=(k&0x55555555)<<1|(k&0xAAAAAAAA)>>1;\
k=(k&0x33333333)<<2|(k&0xCCCCCCCC)>>2;\
k=(k&0x0F0F0F0F)<<4|(k&0xF0F0F0F0)>>4;\
k=(k&0x00FF00FF)<<8|(k&0xFF00FF00)>>8;\
k=(k&0x0000FFFF)<<16|(k& 0xFFFF0000)>>16;\
k>>=(32-L); }
```

Для исполнения БПФ множества комплексных векторов длиной N, уложенных в матрицу размером M×J, преобразуем функцию MFFT, представленную ранее в [7]. Для этого дополним заголовок функции параметром D, а внутрь двух вложенных циклов for (по m и j) поместим вызов модифицированной функции FFT:

```
void MFFT(int M,int J,int N,int D,
complex *X,complex *Y,complex *W){
int m,j,mj;
 for (m=0; m<M; m++)
 for (j=0;j<J;j++){ mj=(m*J+j)*N;
 FFT(N,&X[mj],&Y[mj],W,D);
 }//for j,m
}//MFFT
```

Такая функция позволяет выполнять множественное прямое БПФ (при D=1):

$$Y_{M\times J} = MFFT^{N}(X_{M\times J})$$

или обратное БПФ (при D=0):

$$Y_{M\times J} = iMFFT^{N}(X_{M\times J})$$

с комплексными векторами длиной N, являющимися элементами матрицы X[M][J], помещая результат в матрицу Y[M][J] комплексных векторов, используя при этом комплексный вектор W в качестве таблицы весовых коэффициентов W[0:N]. В ней в теле внутреннего цикла (по j) осуществляется одиночная операция БПФ над векторами, расположенными начиная с позиции mj=(m*J+j)*N в матрицах X и Y.

Помимо операции множественного БПФ для осуществления операции множественной быстрой свёртки потребуется ещё и операция покомпонентного умножения комплексных векторов, являющихся элементами соответствующих матриц размером M×J. Оформим такую операцию в виде следующей функции **MMul**:

```
void MMul(int M,int J,int N,
complex *X,complex *Y,complex *Z){
int m,j,mj,n;
 for (m=0; m<M; m++)
 for (j=0;j<J;j++){ mj=(m*J+j)*N;
 for (n=0; n<N; n++)
 Z[mj+n]= X[mj+n]*Y[mj+n];
 }//for j,m
}//MMul
```

Используя приведённые функции MFFF и MMul, представим теперь в виде функции MFSvrtka алгоритм исполнения операции быстрой свёртки для множества векторов, являющихся элементами соответствующих матриц:

```
void MFSvrtka(int M,int J,int N,
complex *X,complex *Y,complex *W,
complex *G, complex *Z){
 MFFT(M,J,N,1,X,Z,W); // Z=MFFT(X)
 MFFT(M,J,N,1,Y,G,W); // G=MFFT(Y)
 MMul(M,J,N,G,Z,G);  // G=MMul(G,Z)
 MFFT(M,J,N,0,G,Z,W); // Z=iMFFT(G)
}//MFSvrtka
```

Для этой функции в качестве исходных задаются комплексные вектора длиной N как элементы матриц X и Y размером M×J, а также подготовленная заранее таблица весовых коэффициентов Фурье W[0:N]. Результирующие вектора помещаются в матрицу Z[M][J][N], а матрица G[M][J][N] используется в качестве вспомогательной для сохранения промежуточных результатов преобразования.

Такой последовательный алгоритм выполнения операции множественной быстрой свёртки примем в качестве базового. Далее будем рассматривать различные варианты исполнения той же операции множественной свёртки в среде параллельных исполнителей. А также будем сравнивать каждый из них (по производительности) с этим базовым, чтобы выявить, какой вариант даёт максимальное ускорение в среде OpenCL.

3. Первоначальный вариант OpenCL-программы быстрой множественной свёртки

В качестве первоначального варианта (A) предложим такую OpenCL-программу, которая задействует M×J исполнителей OpenCL-среды для исполнения множественной операции свёртки параллельно сразу над всеми парами векторов, взятых на соответствующих позициях из матриц X и Y. При этом для обработки векторов применим тот же план действий, который ис-

пользовался в базовой программе, но будем выполнять его не последовательно, а параллельно для всех пар векторов.

Будем полагать, что в основной OpenCL-программе уже предусмотрены типовые подготовительные действия, необходимые для инициализации OpenCL-среды. Определена OpenCL-платформа и дескрипторы OpenCL-устройств, подготовлен OpenCL-контекст **cntxt** и в нём создан дескриптор очереди команд **cmndQ**, а также приготовлены все объекты **clX**, **clY**, **clZ**, **clG**, **clW** для представления в памяти OpenCL-устройства данных, подлежащих обработке, и особый объект **prgrm**, содержащий в откомпилированной форме код всех процедур ядра. (Как обеспечить все эти действия, было подробно описано ранее в пп. 2.3, 2.4, 2.5 в [3]).

Будем также считать, что в глобальной памяти OpenCL-устройства уже приготовлена таблица весовых коэффициентов Фурье, доступная как объект **clW** (типа cl_mem).

И далее представим процедуры ядра и функции OpenCL-программы, которые обеспечат весь комплекс действий по подготовке и запуску M×J параллельных вычислительных узлов OpenCL-среды для выполнения множественной операции быстрой свёртки.

```
void OpenCLMFSvrtka
(int M, int J, int N,complex *X,
complex *Y,complex *G,complex *Z){
 OpenCLMFFT(M,J,N,X,Z,1);
 OpenCLMFFT(M,J,N,Y,G,1);
 OpenCLMMul(M,J,N,G,Z,G);
 OpenCLMFFT(M,J,N,G,Z,0); }
//OpenCL_MFSvrtka
```

Функции OpenCLMFFT и OpenCLMMul, вызываемые здесь в теле основной функции OpenCLMFSvrtka, предназначены для исполнения в OpenCL-среде операций множественного БПФ и покомпонентного умножения, аналогичные тем, что были рассмотрены ранее в последовательной программе.

Функция **OpenCLMMul** сначала загружает исходные данные матриц Z и Y в память OpenCL-устройства и затем вызывает функцию **clMMul**, которая запускает на каждом вычислительном узле устройства процедуру ядра для исполнения операции покомпонентного умножения назначенной ей пары векторов.

```
void OpenCLMMul(int M,int J,int N,
complex *Z,complex *Y,complex *G){
// write Z matrix into clZ of device memory
cl_uint szC= sizeof(complex);
clEnqueueWriteBuffer(cmndQ,clZ,
 CL_TRUE,0,M*J*N*szC,Z,0,0,0);
// write Y matrix into clY of device memory
clEnqueueWriteBuffer(cmndQ,clY,
 CL_TRUE,0,M*J*N*szC,Y,0,0,0);
// perform clG=MMul(clZ,clY) in OpenCL-device
clMMul(M,J,N,&clZ,&clY,&clG);
// read the result from clG to the Host in matrix G
clEnqueueReadBuffer(cmndQ,clG,
 CL_TRUE,0,M*J*N*szC,G,0,0,0);
}//OpenCLMMul
void clMMul(int M, int J, int N,
cl_mem *pZ,cl_mem *pY,cl_mem *pG){
cl_uint szI= sizeof(int);
cl_uint szM= sizeof(cl_mem);
clSetKernelArg(knMM,0,szI,&M);
clSetKernelArg(knMM,1,szI,&J);
clSetKernelArg(knMM,2,szI,&N);
clSetKernelArg(knMM,3,szM,pZ);
clSetKernelArg(knMM,4,szM,pY);
clSetKernelArg(knMM,5,szM,pG);
size_t gWS[2]={M,J};cl_event evMM;
clEnqueueNDRangeKernel(cmndQ,
 knMM,2,NULL,gWS,NULL,0,0,&evMM);
clFinish(cmndQ);
}//clMMul
```

А после исполнения такой операции над всеми парами векторов сформированная в OpenCL-памяти результирующая матрица, представленная дескриптором clG, уже копируется в матрицу G основной памяти. Саму же операцию покомпонентного умножения элементов двух векторов в памяти OpenCL-устройства исполняет процедура ядра, которая запускается параллельно на всех M×J узлах OpenCL-устройства:

```
__kernel void kernMMul
( uint M, uint J, uint N,
const __global complex *Z,
 const __global complex *Y,
  __global complex *G ) {
 int m,j,mj, i;
 m= get_global_id(0);
 j= get_global_id(1);
 mj=(m*J+j)*N;
 for (i=0; i<N; i++)
 G[mj+i]= Z[mj+i]*Y[mj+i];
}//kernMMul
```

В OpenCL-программе процедура ядра **kernMMul** представлена своим дескриптором **knMM**, который должен быть предварительно проинициализирован после компиляции программы ядра, приготовленной на языке OpenCL (в файле “MFSvrtka.cl”):

```
knMM=clCreateKernel(prgrm,"kernMMul",0);
```

Аналогично следует проинициализировать и дескриптор **knMF**:

```
knMF=clCreateKernel(prgrm,"kernMFFT",0);
```

который будет представлять в OpenCL-программе процедуру ядра **kernMFFT**, предназначенную для исполнения вычислительными

узлами OpenCL-устройства множественной операции БПФ:

```
__kernel void kernMFFT
( uint M, uint J, uint N, uint D,
 const __global complex *X,
  __global complex *Y,
 const __global complex *W ) {
 int m,j,mj, i; complex XP[NP];
 int P=iLog2(N);
 m= get_global_id(0);
 j= get_global_id(1);
 mj=(m*J+j)*N;
//1. Бит-реверсная перестановка:
 copy_brvprst(N,P,&X[mj],XP);
//2.Основной цикл исполнения бабочек Фурье:
 FFTMainLoopP(N,P,XP,W,D);
if(D)// Нормализация (для обратного БПФ)
 for(i=0;i<N;i++) Y[mj+i]=XP[i];
else { float Norm=1.00/(float)N;
 for (i=0;i<N;i++)
 Y[mj+i]=cMScal(XP[i],Norm);
}//if D
}//kernMFFT
```

В этой процедуре используются две вспомогательные функции, которые оформлены отдельно. Одна из них **copy_brvprst** осуществляет бит-реверсное копирование исходного вектора, взятого с позиции mj матрицы X в вектор XP, который располагается в быстродействующей локальной (private) памяти исполнителя:

```
void copy_brvprst (int N, int L,
__global complex *X,complex *XP){
 unsigned int i,j;
 for (i=0; i<N; i++) {
 BRvrsVal(j,i,L); XP[j]=X[i];
 }// for i
}//copy_brvprst
```

А другая функция **FFTMainLoopP** реализует основной цикл исполнения операций “бабочек Фурье” над элементами вектора XP:

```
void FFTMainLoopP(uint N, uint P,
 complex *XP,__global complex *W,
 int D){ complex V;
int t,s,h,G,R,d,k,u,a,b;
s=1;h=2;G=1;R=N/2;d=N/2;
for (t=1; t<=P; t++) {
 for(k=0,u=0;k<G;k++,u=u+d){
 if(D) V=W[u];else V=W[N-u];
 for(i=0,a=k; i<R; i++,a=a+h)
 {b=a+s; BF(Y[a],Y[b],V);}
 }//for k
 h=h*2;s=s*2;G=G*2;R=R/2;d=d/2;
}//for t
}//FFTMainLoopP
```

А завершается процедура **kernMFFT** копированием полученного вектора XP на своё место в матрицу Y (с позиции mj) с нормализацией элементов для обратного БПФ (при D=0).

Для запуска процедуры ядра **kernMFFT** параллельно на всех M×J вычислительных узлах OpenCL-устройства в OpenCL-программе вызывается функция **clMFFT** :

```
void clMFFT(int M, int J, int N,
 cl_mem *pX,cl_mem *pY, int D ){
cl_uint szI= sizeof(int);
cl_uint szM= sizeof(cl_mem);
clSetKernelArg(knMF,0,szI,&M);
clSetKernelArg(knMF,1,szI,&J);
clSetKernelArg(knMF,2,szI,&N);
clSetKernelArg(knMF,3,szI,&D);
clSetKernelArg(knMF,4,szM,pX);
clSetKernelArg(knMM,5,szM,pY);
clSetKernelArg(knMM,6,szM,&clW);
size_t gWS[2]={M,J};cl_event evMF;
clEnqueueNDRangeKernel(cmndQ,
 knMF,2,NULL,gWS,NULL,0,0,&evMF);
clFinish(cmndQ);
}//clMFFT
```

После исполнения каждым узлом процедуры **kernMFFT** в памяти OpenCL-устройства в матрице, задаваемой указателем **pY**, формируется результат множественной операции БПФ над соответствующими векторами, взятыми из матрицы, задаваемой указателем **pX**.

Функция **clMFFT** вызывается из тела функции **OpenCLMFFT** после того, как та запишет заданную ей в качестве параметра матрицу **X** в память OpenCL-устройства, выделенную под дескриптор **clX**. А после такого вызова матрицу, полученную как результат множественной операции БПФ, функция **OpenCLMFFT** перекопирует из OpenCL-памяти, выделенной под дескриптор **clZ**, в память OpenCL-программы по месту, задаваемому параметром **Z**.

```
void OpenCLMFFT(int M,int J,int N,
 complex *X,complex *Z, int D) {
// write X matrix into clX of device memory
cl_uint szC= sizeof(complex);
clEnqueueWriteBuffer(cmndQ,clX,
 CL_TRUE,0,M*J*N*szC,X,0,0,0);
// perform clZ=MFFT(clX) in OpenCL-device
clMFFT(M,J,N,&clX,&clZ,D);
// read the result from clZ to the Host in matrix Z
clEnqueueReadBuffer(cmndQ,clZ,
 CL_TRUE,0,M*J*N*szC,Z,0,0,0);
}//OpenCLMFFT
```

Заметим, что функция **OpenCLMFFT** вызывается из основной функции **OpenCLMFSvrtka** аж 3 раза. Первый раз, чтобы получить в матрице Z результат множественного прямого БПФ, произведённого в OpenCL-среде над векторами матрицы X. Второй раз, чтобы получить во вспомогательной матрице G результат множественного

прямого БПФ над векторами матрицы Y. А третий раз, чтобы получить итоговую матрицу Z как результат множественного обратного БПФ над векторами вспомогательной матрицы G, изменённой после её покомпонентного перемножения с векторами матрицы Z.

Теперь попробуем оценить, насколько удаётся ускорить операцию множественной быстрой свёртки полученным вариантом (А) OpenCL-программы. Для этого сравним время исполнения ($\mathbf{T_{CL}}$) функции **OpenCLMFSvrtka** (в среде OpenCL с множеством параллельных исполнителей) со временем исполнения ($\mathbf{T_{CPU}}$) функции **MFSvrtka** (на одном процессоре) с теми же параметрами и вычислим коэффициент ускорения **K** как отношение $\mathbf{T_{CPU}/T_{CL}}$.

Программа с вызовами этих функций была разработана (на языке Си) как консольное приложение в MS Visual Studio. Она компоновалась и многократно прогонялась на различных OpenCL-платформах для матриц комплексных векторов одинарной точности различных размеров (M×J) и длин (N). Усреднённые показатели коэффициентов ускорения, полученные в результате таких прогонов, представлены в таблицах 1 и 2.

Таблица 1 демонстрирует показатели ускорения, полученные на аппаратной платформе, оснащённой процессором CPU Intel-i59400 с частотой 2.9 Ггц и встроенным графическим процессором GPU UHD 630 с частотой 350 МГц (будем называть её платформа «Intel»).

Таблица 1. Ускорение множественной свёртки OpenCL-программой варианта А на платформе Intel

M×J: N	**2×2**	**5×5**	**10 ×10**	**20 ×20**	**50 ×50**	**100 ×100**	**200 ×200**
32	0.148	0.841	3.405	11.920	**29.990**	24.522	22.771
64	0.273	1.501	5.206	16.855	26.474	22.070	20.911
128	0.360	2.315	8.192	**22.874**	22.561	19.963	19.905
256	0.468	2.649	10.749	24.742	21.691	19.392	20.897
512	0.548	2.964	10.244	13.163	4.698	4.545	---
1024	0.600	3.539	10.498	10.034	4.396	3.970	---
2048	0.613	2.855	9.664	8.801	3.904	---	---
4096	0.641	3.012	9.181	7.070	3.944	---	---
8192	0.670	3.101	8.320	6.311	---	---	---
16384	0.645	3.229	6.590	5.992	---	---	---
32768	0.648	2.635	5.764	5.611	---	---	---
65536	0.661	2.199	6.379	---	---	---	---

Замечание: символы --- в таблице 1означают, что для таких значений параметров программа не может быть корректно исполнена (например, из-за недостатка памяти).

А таблица 2 представляет показатели ускорения, полученные на аппаратной платформе, содержащей процессор Intel i3-2100 с частотой 3.1 Ггц и специализированную видеокарту NVidia GeForce 1050ti с частотой 1392 Мгц (будем называть её платформа «NVidia»).

Таблица 2. Ускорение множественной свёртки OpenCLпрограммой варианта А на платформе NVidia

M×J: N	**2×2**	**5×5**	**10 ×10**	**20 ×20**	**50 ×50**	**100 ×100**	**200 ×200**
32	0.116	0.787	2.881	9.414	30.250	32.962	36.077
64	0.337	1.802	6.281	17.694	34.965	40.192	42.344
128	0.539	3.640	15.920	31.400	42.473	48.575	49.209
256	0.727	4.955	11.526	31.545	54.953	52.125	52.175
512	1.206	6.438	16.857	36.950	64.714	60.089	58.099
1024	1.429	6.716	18.811	44.194	66.867	64.929	61.224
2048	1.526	7.287	24.404	47.576	77.568	69.470	---
4096	1.499	7.874	23.853	56.910	82.800	75.480	---
8192	1.454	8.210	26.761	59.489	**84.016**	---	---
16384	1.499	8.515	26.673	77.436	**100.41**	---	---
32768	1.395	8.121	24.688	65.389	---	---	---
65536	1.168	6.290	21.880	33.502	---	---	---

Заметим, что в этом варианте А программы множественной свёртки был заимствован не самый оптимальный вариант реализации операции множественного БПФ, который был представлен ранее в [7, п.4]. Но сравнивая показатели ускорения операции множественного БПФ этого варианта В, представленные в таблицах 7 и 3 в [7], с показателями ускорения операции множественной свёртки из таблиц 1 и 2, можно заметить, что и на платформе “Intel”, и на платформе NVidia почти для всех параметров M,J,N операцию множественной свёртки не удалось ускорить лучше, чем саму операцию множественного БПФ. Так, например, на платформе NVidia на параметрах M×J=50×50 N=8192 для операции множественного БПФ удалось достичь ускорения в 125 раз, а для операции множественной свёртки – лишь в 84 раза.

Возникает вопрос, а можно ли вообще ускорить по технологии OpenCL операцию быстрой свёртки в лучшей степени, чем саму операцию БПФ, которая применяется для её исполнения? Оказывается, можно. Рассмотрим следующие варианты OpenCL-программы, которые позволяют этого добиться.

4. Варианты оптимизации OpenCL-программы быстрой множественной свёртки

Заметим, что в OpenCL-программе варианта А приходится многократно передавать блоки данных, представляющие матрицы комплексных чисел размером M×J×N, из основной памяти в память OpenCL-устройства и обратно. При каждом вызове функции **OpenCLMFFT** такая передача выполняется дважды, а при вызове функции **OpenCLMMul** ─ трижды. Итого для одного исполнения функции **OpenCLMFSvrtka** потребуется аж 9 таких передач блоков данных.

Попытаемся модифицировать функцию множественной свёртки так, чтобы минимизировать

количество таких передач. Для этого все матрицы, в которых формируются промежуточные результаты, будем просто оставлять в памяти OpenCL-устройства, не копируя их в основную память. И затем сразу же их использовать при последующей обработке процедурами ядра.

Выразим такие изменения в OpenCL-программе, оформив другую функцию для исполнения множественной свёртки:

```
void OpenCLMFSvrtkaB
(int M, int J, int N,complex *X,
complex *Y,complex *G,complex *Z){
// write X matrix into clX of device memory
cl_uint szC= sizeof(complex);
clEnqueueWriteBuffer(cmndQ,clX,
 CL_TRUE,0,M*J*N*szC,X,0,0,0);
// write Y matrix into clY of device memory
clEnqueueWriteBuffer(cmndQ,clY,
 CL_TRUE,0,M*J*N*szC,Y,0,0,0);
// perform clZ=MFFT(clX) in OpenCL-device
clMFFT(M,J,N,&clX,&clZ,1);
// perform clG=MFFT(clY) in OpenCL-device
clMFFT(M,J,N,&clY,&clG,1);
// perform clG=MMul(clG,clZ) in OpenCL-device
clMMul(M,J,N,&clG,&clZ,&clG);
// perform clZ=iMFFT(clG) in OpenCL-device
clMFFT(M,J,N,&clG,&clZ,0);
// read the result from clZ to the Host in matrix Z
clEnqueueReadBuffer(cmndQ,clZ,
 CL_TRUE,0,M*J*N*szC,Z,0,0,0);
//OpenCL_MFSvrtkaB
```

В результате такой модификации количество передач матричных блоков данных для одной операции свёртки сократится с девяти до трёх. Посмотрим, как это отразится на показателях, характеризующих ускорение операции множественной свёртки в среде OpenCL.

Результаты полученного нового варианта (B) OpenCL-программы представлены в таблицах 3 и 4 (для платформ «Intel» и «NVidia»).

Можно отметить, что показатели ускорения в этих таблицах почти для всех параметров M,J,N заметно улучшились (по сравнению с вариантом A). Причём теперь коэффициенты ускорения операции множественной свёртки во многих случаях даже превосходят коэффициенты ускорения множественной операции БПФ (из таблиц 7 и 3 в [7]) для данных тех же размеров. Так, например, на параметрах M×J=50×50 N=8192 для платформы NVidia коэффициент ускорения множественной операции свёртки вырос до значения **151** и тем самым превзошёл значение 125, которое было отмечено для операции множественного БПФ (в таблице 3 в [7]).

Таблица 3. Ускорение множественной свёртки OpenCL-программой варианта B на платформе Intel

M×J:	**2×2**	**5×5**	**10**	**20**	**50**	**100**	**200**
N			**×10**	**×20**	**×50**	**×100**	**×200**
32	0.153	0.919	3.733	13.372	37.219	39.701	**40.897**
64	0.282	1.586	6.002	21.210	38.644	32.144	30.269
128	0.355	2.398	8.755	26.735	30.126	26.496	26.554
256	0.468	2.649	10.749	24.742	21.691	19.392	28.145
512	0.549	2.943	10.547	14.664	4.757	4.706	---
1024	0.592	3.551	10.865	10.882	4.534	4.094	---
2048	0.632	2.991	8.946	9.431	3.995	---	---
4096	0.615	3.049	9.558	7.894	4.025	---	---
8192	0.661	3.114	8.433	6.253	---	---	---
16384	0.643	3.273	6.721	6.206	---	---	---
32768	0.649	2.661	5.930	5.961	---	---	---
65536	0.660	2.237	6.548	---	---	---	---

Таблица 4. Ускорение множественной свёртки OpenCLпрограммой варианта B на платформе NVidia

M×J: N	**2×2**	**5×5**	**10 ×10**	**20 ×20**	**50 ×50**	**100 ×100**	**200 ×200**
32	0.233	1.308	5.000	17.063	65.154	65.923	74.053
64	0.500	2.646	11.167	35.389	71.179	75.075	77.631
128	0.710	4.440	26.534	52.333	93.440	80.958	90.250
256	0.941	6.812	19.909	57.833	104.41	94.773	99.606
512	1.708	8.822	24.842	67.182	104.10	107.89	104.74
1024	1.818	8.569	26.237	72.318	132.06	106.90	112.54
2048	1.723	8.132	31.888	80.052	139.44	119.99	---
4096	1.578	8.574	29.346	83.549	137.85	129.63	---
8192	1.490	8.794	31.208	86.613	**151.05**	---	---
16384	1.517	9.042	31.106	111.78	**174.67**	---	---
32768	1.416	8.488	27.812	90.039	---	---	---
65536	1.175	6.455	23.852	38.298	---	---	---

Опробуем ещё один возможный путь оптимизации OpenCL-программы. Он связан с минимизацией накладных расходов, требующихся на запуск процедур ядра. В предыдущем варианте B требовалось обеспечить 4 таких запуска. Попробуем сосредоточить все действия, которые надлежит исполнить каждым вычислительным узлом OpenCL-устройства, в одной единственной процедуре ядра. Такая процедура будет запускаться один раз (на всех узлах сразу) и должна будет исполнить весь комплекс действий с назначенными ей векторами из заданных матриц, требующийся для получения свёртки.

```
__kernel void kernMFSvrtka
( uint M, uint J, uint N,
 const __global complex *X,
 const __global complex *W,
  __global complex *Y,
  __global complex *Z ) {
unsigned int m,j,mj, i,q;
complex V, T;
complex XP[NP]; complex YP[NP];
const __global complex *VX;
__global complex *VY,*VZ;
int P=iLog2(N);
m=get_global_id(0);
j=get_global_id(1); mj=(m*J+j)*N;

VX=&X[mj]; VY=&Y[mj]; VZ=&Z[mj];
//1. Прямое БПФ: XP=FFT(VX); YP=FFT(VY);
```

```
//1a. Бит-реверсное копир. VX=>XP; VY=>YP;
for(i=0;i<N;i++){ BRvrsVal(q,i,P);
 XP[q]=VX[i]; YP[q]=VY[i];}//for i
//1b.Основной цикл исполнения бабочек Фурье:
int t,s,h,G,R,d,k,u,a,b;
s=1;h=2;G=1;R=N/2;d=N/2;
for (t=1; t<=P; t++) {
for(k=0,u=0;k<G;k++,u=u+d){V=W[u];
 for(i=0,a=k;i<R;i++,a=a+h){b=a+s;
 BF(XP[a],XP[b],V);
 BF(YP[a],YP[b],V); }//for i
}//for k
h=h*2;s=s*2;G=G*2;R=R/2;d=d/2;
}//for t
//2. Поэлем. перемножение: XP=MMul(XP,YP)
for(i=0;i<N;i++)XP[i]=XP[i]*YP[i];
//3. Обратное БПФ: VZ=iFFT(XP);
//3a. Бит-реверсная перестановка вектора XP:
for(i=0;i<N;i++){ BRvrsVal(q,i,P);
 if(q>i)//XP[q]<=>XP[i];
 {T=XP[q]; XP[q]=XP[i]; XP[i]=T;}
}// for i
//3b.Основной цикл исполнения бабочек Фурье:
FFT_MainLoopP(N,P,XP,W,0);
//3c Копирование с нормализацией XP=>VX :
float Norm=1.00/(float)N;
for (i=0;i<N;i++)
 VZ[i]=cMScal(XP[i],Norm);
}//for i
}//kernMFSvrtka
```

В этой процедуре применён ещё один приём оптимизации: выполняется совместное прямое БПФ сразу для двух векторов VX и VY, взятых из исходных матриц X и Y (на позиции mj). Это позволяет не вычислять дважды индексы в цикле бит-реверсного копирования и многочисленные параметры в цикле исполнения операций «бабочек Фурье».

Для новой процедуры ядра **kernMFSvrtka** проинициализируем дескриптор, представляющий её в OpenCL-программе:

```
knMS=clCreateKernel(prgrm,"kernMFSvrtka",0);
```

И составим функцию, которая будет выполнять операцию множественной свёртки, запуская лишь эту единственную процедуру ядра на M×J вычислительных узлах OpenCL-среды:

```
void OpenCLMFSvrtkaC
(int M, int J, int N,complex *X,
complex *Y,complex *G,complex *Z){
// write X matrix into clX of device memory
clEnqueueWriteBuffer(cmndQ,clX,
 CL_TRUE,0,M*J*N*szC,X,0,0,0);
// write Y matrix into clY of device memory
clEnqueueWriteBuffer(cmndQ,clY,
 CL_TRUE,0,M*J*N*szC,Y,0,0,0);
clSetKernelArg(knMS,0,szI,&M);
clSetKernelArg(knMS,1,szI,&J);
clSetKernelArg(knMS,2,szI,&N);
clSetKernelArg(knMS,3,szM,&clX);
clSetKernelArg(knMS,4,szM,&clW);
clSetKernelArg(knMS,5,szM,&clY);
clSetKernelArg(knMS,6,szM,&clZ);
size_t gWS[2]={M,J};cl_event evMS;
clEnqueueNDRangeKernel(cmndQ,
 knMS,2,NULL,gWS,NULL,0,0,&evMS);
clFinish(cmndQ);
// read the result from clZ to the Host in matrix Z
clEnqueueReadBuffer(cmndQ,clZ,
 CL_TRUE,0,M*J*N*szC,Z,0,0,0);
}//OpenCL_MFSvrtkaC
```

В результате такой оптимизации получим ещё один вариант (C) OpenCL-программы для выполнения множественной свёртки в среде OpenCL. Показатели ускорения этой операции (вариантом C) для платформ «Intel» и «NVidia» представим в таблицах 5 и 6 соответственно.

Таблица 5. Ускорение множественной свёртки OpenCL-программой варианта C на платформе Intel

M×J: N	2×2	5×5	10 ×10	20 ×20	50 ×50	100 ×100	200 ×200
32	0.166	0.978	3.890	12.450	38.678	42.702	46.873
64	0.291	1.716	6.074	19.584	36.668	41.656	45.263
128	0.388	2.457	8.519	25.621	38.743	41.033	46.240
256	0.521	2.973	11.210	29.956	16.262	13.731	15.549
512	0.613	3.217	11.033	12.008	4.880	4.862	---
1024	0.659	3.709	11.132	10.132	4.572	4.241	---
2048	0.624	3.131	9.648	8.322	4.023	---	---
4096	0.701	2.781	9.610	7.158	4.025	---	---
8192	0.656	3.244	7.718	6.642	---	---	---
16384	0.688	3.131	6.326	6.100	---	---	---
32768	0.658	2.426	6.169	5.596	---	---	---
65536	0.680	2.156	6.065	---	---	---	---

Таблица 6. Ускорение множественной свёртки OpenCLпрограммой варианта C на платформе NVidia

M×J: N	2×2	5×5	10 ×10	20 ×20	50 ×50	100 ×100	200 ×200
32	0.400	2.576	8.095	27.300	77.000	81.619	93.800
64	0.814	3.943	16.080	37.471	79.720	88.422	98.803
128	1.000	7.114	31.841	52.333	116.80	107.95	121.13
256	1.311	8.385	24.333	69.400	116.01	115.83	117.39
512	1.864	9.528	27.765	67.182	126.02	131.86	128.00
1024	1.818	8.283	27.694	75.762	135.45	139.26	131.29
2048	1.513	8.502	32.319	78.349	148.26	140.24	---
4096	1.596	8.929	31.300	84.426	146.57	149.87	---
8192	1.480	10.452	35.873	103.49	**154.82**	---	---
16384	1.530	9.196	31.495	111.15	---	---	---
32768	1.781	10.298	31.106	65.087	---	---	---
65536	1.205	6.635	23.705	---	---	---	---

Как видно из приведённых таблиц, для многих наборов параметров (M,J,N) вариант C OpenCL-программы обеспечивает и на платформе Intel, и на платформе NVidia наилучшие показатели ускорения среди всех уже рассмотренных нами вариантов (A,B,C).

5. Оптимизация множественной свёртки комплексных векторов большой длины

До сих пор для выполнения множественной свёртки применялся простой, но не самый оптимальный вариант множественной операции БПФ, описанный в [7,п.4] как вариант B. Ранее был предложен более оптимальный способ исполнения операции множественного БПФ в среде OpenCL, описанный в [7,п.5] как вариант C. Он позволяет существенно ускорить такую операцию для комплексных векторов большой длины. Попробуем применить такой же вариант ускорения прямого и обратного множественного БПФ для дальнейшей оптимизации множественной свёртки.

Процедуры ядра множественного БПФ, представленные в [7,п.5] модифицируем так, чтобы приспособить их для исполнения как прямого (при D=1), так и обратного БПФ (при D=0) ансамблем из N×M×J параллельных исполнителей. Для этого переделаем процедуру бит-реверсного копирования элемента:

```
__kernel void kernMFFTbp
( uint N, uint L,
 const __global complex *X,
  __global complex *Y,
 uint M, uint J ) {
 uint m,j,mj, k,i;
 i= get_global_id(0);
 m= get_global_id(1);
 j= get_global_id(2);
 mj=(m*J+j)*N;
 BRvrsVal(k,i,L);
 Y[mj+k]= X[mj+i];
}//kernMFFTbp
```

А также процедуру исполнения операции «бабочки Фурье» для пары элементов:

```
__kernel void kernMFFTbf
( uint t, uint N,
 __global complex *Y,
 __global complex *W,
 uint M, uint J, uint D) {
 uint gm,gj,mj, gi; complex V;
 uint s,h,G,R,k,u,j,a,b;
 gi= get_global_id(0);
 gm= get_global_id(1);
 gj= get_global_id(2);
 mj=(gm*J+gj)*N;
G=1<<(t-1);R=N>>t;// G=2^(t-1),R=2^(P-t);
k=gi&(G-1);j=gi>>(t-1);//k=gi%G,j=gi/G
 s=G;h=1<<t; a=k+j*h;b=a+s; u=R*k;
 V=D? W[u]:W[N-u] /*Conj(W[u])*/;
 BF(Y[mj+a],Y[mj+b],V);}
}//kernMFFTbf
```

И добавим процедуру нормализации элемента:

```
__kernel void kernMFFTnm
( uint N, uint M, uint J,
__global complex *X, float Norm){
 uint i= get_global_id(0);
 uint m= get_global_id(1);
 uint j= get_global_id(2);
 uint mj=(m*J+j)*N;
 X[mj+i]=cMScal(X[mj+i],Norm);
}//kernMFFTnm
```

В OpenCL-программе проинициализируем дескрипторы этих процедур:

```
knMp=clCreateKernel(prgrm,"kernMFFTbp",0);
knMb=clCreateKernel(prgrm,"kernMFFTbf",0);
knMn=clCreateKernel(prgrm,"kernMFFTnm",0);
```

И предусмотрим функцию, которая будет запускать эти процедуры ядра параллельно на N×M×J вычислительных узлах OpenCL-среды, чтобы выполнить операцию множественного БПФ сразу над всеми векторами, уложенными в памяти OpenCL-устройства в матрицы, заданные в параметрах функции указателями на дескрипторы (pX,pY).

```
void clMFFTD(int M, int J, int N,
 cl_mem *pX,cl_mem *pY, int D ){
cl_uint szI= sizeof(int);
cl_uint szM= sizeof(cl_mem);
cl_uint szF=sizeof(float);
cl_event evMp,evMb,evMn;
size_t gWS[3]={N,M,J};
uint P=iLog2(N);
// 1. Bit-reverse copyng X[m][j] => Y[m][j]
clSetKernelArg(knMp,0,szI,&N);
clSetKernelArg(knMp,1,szI,&P);
clSetKernelArg(knMp,2,szM,pX);
clSetKernelArg(knMp,3,szM,pY);
clSetKernelArg(knMp,4,szI,&M);
clSetKernelArg(knMp,5,szI,&J);
clEnqueueNDRangeKernel(cmndQ,
 knMp,3,NULL,gWS,NULL,0,0,&evMp);
clFinish(cmndQ);
// 2. Main Loop for execution butterfly operations:
clSetKernelArg(knMb,1,szI,&N);
clSetKernelArg(knMb,2,szM,pY);
clSetKernelArg(knMb,3,szM,&clW);
clSetKernelArg(knMb,4,szI,&M);
clSetKernelArg(knMb,5,szI,&J);
clSetKernelArg(knMb,6,szI,&D);
gWS[0]= N/2;
for (int t=1; t<=P; t++) {
 clSetKernelArg(knMb,0,szI,&t);
 clEnqueueNDRangeKernel(cmndQ,
 knMF,3,NULL,gWS,NULL,0,0,&evMb);
 clWaitForEvents(1,&evMb);
}//for t
clFinish(cmndQ);
// 3. Normalization (for InDirect FFT)
```

```
if(!D) { gWS[0]=N;
 float Norm=1.00/(float)N;
 clSetKernelArg(knMn,0,szI,&N);
 clSetKernelArg(knMn,1,szI,&M);
 clSetKernelArg(knMn,2,szI,&J);
 clSetKernelArg(knMn,3,szM,pY);
 clSetKernelArg(knMn,4,szF,&Norm);
 clEnqueueNDRangeKernel(cmndQ,
 knMn,3,NULL,gWS,NULL,0,0,&evMn);
 clFinish(cmndQ); }//if
}//clMFFTD
```

Используя её, составим для нового варианта (D) OpenCL-программы функцию выполнения множественного БПФ над M×J векторами, уложенными в заданные матрицы:

```
void OpenCLMFFTD(int M,int J,int N,
 complex *X,complex *Z, int D) {
// write X matrix into clX of device memory
clEnqueueWriteBuffer(cmndQ,clX,
 CL_TRUE,0,M*J*N*szC,X,0,0,0);
// perform clZ=MFFT(clX) in OpenCL-device
clMFFTD(M,J,N,&clX,&clZ,D);
// read the result from clZ to the Host in matrix Z
clEnqueueReadBuffer(cmndQ,clZ,
 CL_TRUE,0,M*J*N*szC,Z,0,0,0);
}//OpenCLMFFTD
```

Такая функция **OpenCLMFFTD** записывает исходную матрицу векторов, заданную как параметр X, из основной памяти в память OpenCL-устройства, затем вызывает функцию **clMFFTD** для исполнения на устройстве множественного БПФ сразу над всеми векторами, а после копирует полученный результат из памяти устройства в основную память по месту расположения матрицы, заданной параметром Z.

Аналогичную оптимизацию проделаем и для множественной операции покомпонентного умножения, используя для её исполнения в OpenCL-среде не M×J, а N×M×J вычислителей.

Для этого определим новую процедуру ядра, в которой обработка всех N элементов векторов выполняется не одним исполнителем в последовательном цикле, а параллельно сразу N исполнителями, каждый из которых получает результат перемножения для своего элемента:

```
__kernel void kernMMulD
( uint M, uint J, uint N,
 const __global complex *Z,
 const __global complex *Y,
  __global complex *G ) {
 int i= get_global_id(0);
 int m= get_global_id(1);
 int j= get_global_id(2);
 int mj=(m*J+j)*N;
 G[mj+i]= Z[mj+i]*Y[mj+i];
}//kernMMulD
```

В OpenCL-программе проинициализируем дескриптор новой процедуры и модифицируем соответствующие функции:

```
knMm=clCreateKernel(prgrm,"kernMMulD",0);
```

```
void clMMulD(int M, int J, int N,
cl_mem *pZ,cl_mem *pY,cl_mem *pG){
cl_uint szI= sizeof(int);
cl_uint szM= sizeof(cl_mem);
clSetKernelArg(knMm,0,szI,&N);
clSetKernelArg(knMm,1,szI,&M);
clSetKernelArg(knMm,2,szI,&J);
clSetKernelArg(knMm,3,szM,pZ);
clSetKernelArg(knMm,4,szM,pY);
clSetKernelArg(knMm,5,szM,pG);
size_t gWS[3]={N,M,J};
cl_event evMm;
clEnqueueNDRangeKernel(cmndQ,
 knMm,3,NULL,gWS,NULL,0,0,&evMm);
clFinish(cmndQ);
}//clMMulD
```

```
void OpenCLMMulD(int M,int J,int N,
complex *Z,complex *Y,complex *G){
// write Z matrix into clZ of device memory
cl_uint szC= sizeof(complex);
clEnqueueWriteBuffer(cmndQ,clZ,
 CL_TRUE,0,M*J*N*szC,Z,0,0,0);
// write Y matrix into clY of device memory
clEnqueueWriteBuffer(cmndQ,clY,
 CL_TRUE,0,M*J*N*szC,Y,0,0,0);
// perform clG=MMul(clZ,clY) in OpenCL-device
clMMulD(M,J,N,&clZ,&clY,&clG);
// read the result from clG to the Host in matrix G
clEnqueueReadBuffer(cmndQ,clG,
 CL_TRUE,0,M*J*N*szC,G,0,0,0);
}//OpenCLMMulD
```

Используя оптимизированные варианты функций **OpenCLMFFTD** и **OpenCLMMulD**, составим новый вариант (D) функции для исполнения операции множественной свёртки в среде OpenCL:

```
void OpenCLMFSvrtkaD
(int M, int J, int N,complex *X,
complex *Y,complex *G,complex *Z){
 OpenCLMFFTD(M,J,N,X,Z,1);
 OpenCLMFFTD(M,J,N,Y,G,1);
 OpenCLMMulD(M,J,N,G,Z,G);
 OpenCLMFFTD(M,J,N,G,Z,0);
}//OpenCL_MFSvrtkaD
```

В результате проведённой оптимизации получим вариант (D) OpenCL-программы для выполнения множественной свёртки в среде OpenCL, который позволяет значительно ускорить такую операцию для векторов большой длины. Это подтверждается улучшенными коэффициентами ускорения варианта D, представленными в таблицах 7 и 8 для платформ «Intel» и «NVidia» соответственно, по сравнению с теми

же коэффициентами варианта A (из таблиц 1 и 2), по крайней мере, для всех наборов параметров M,J,N при N≥512.

Таблица 7. Ускорение множественной свёртки OpenCL-программой варианта D на платформе Intel

M×J: N	2×2	5×5	10 ×10	20 ×20	50 ×50	100 ×100	200 ×200
32	0.043	0.266	1.077	3.978	16.483	26.564	22.720
64	0.091	0.548	2.101	7.148	23.940	28.269	21.207
128	0.172	1.165	4.053	12.510	28.755	27.373	22.960
256	0.354	2.201	7.887	19.826	33.537	25.227	24.365
512	0.696	4.088	13.143	29.752	32.775	29.392	---
1024	1.405	8.227	22.293	33.875	30.287	25.662	---
2048	2.204	12.200	27.474	38.430	31.317	---	---
4096	5.027	16.844	37.649	32.421	30.034	---	---
8192	7.178	37.084	43.632	30.501	---	---	---
16384	17.38	49.275	31.167	28.455	---	---	---
32768	28.33	**48.156**	37.050	34.616	---	---	---
65536	37.28	39.669	34.137	---	---	---	---

Заметим, однако, что на платформе «NVidia» на некоторых наборах параметров даже при больших значениях N показатели варианта D уступают показателям варианта B (и тем более C). Так, например, для M×J=50×50 и N=8192 OpenCL-программа варианта D даёт коэффициент ускорения 93.440, что лучше, чем значение 84.016 варианта A, но хуже значения 151.05 варианта B (и значения 154.82 варианта C).

Таблица 8. Ускорение множественной свёртки OpenCLпрограммой варианта D на платформе NVidia

M×J: N	2×2	5×5	10 ×10	20 ×20	50 ×50	100 ×100	200 ×200
32	0.063	0.429	1.441	5.151	18.822	30.607	37.520
64	0.164	0.954	3.865	9.800	28.884	38.260	44.361
128	0.277	1.932	13.267	22.429	46.720	51.132	52.490
256	0.640	4.192	14.600	38.556	49.719	54.868	57.667
512	1.367	5.955	24.842	43.471	63.011	65.027	62.493
1024	2.963	14.200	35.607	54.862	74.402	65.736	64.772
2048	4.807	28.774	56.943	68.193	85.494	76.064	---
4096	10.46	40.300	64.561	83.549	84.556	83.597	---
8192	23.47	55.851	79.674	90.359	**93.440**	---	---
16384	28.23	68.284	90.944	121.36	114.12	---	---
32768	44.67	78.675	105.53	113.57	---	---	---
65536	54.03	105.05	113.89	**131.30**	---	---	---

Попробуем улучшить OpenCL-программу варианта D, применив тот же приём оптимизации, какой был применён при превращении программы варианта A в программу варианта B. Для этого постараемся минимизировать количество передач матричных блоков из основной памяти в память OpenCL-устройства и обратно при исполнении операции множественной свёртки в среде OpenCL. Получим следующий вариант (E) функции множественной свёртки:

```
void OpenCLMFSvrtkaE
(int M, int J, int N,complex *X,
complex *Y,complex *G,complex *Z){
// write X matrix into clX of device memory
cl_uint szC= sizeof(complex);
clEnqueueWriteBuffer(cmndQ,clX,
 CL_TRUE,0,M*J*N*szC,X,0,0,0);
// write Y matrix into clY of device memory
clEnqueueWriteBuffer(cmndQ,clY,
 CL_TRUE,0,M*J*N*szC,Y,0,0,0);
// perform clZ=MFFT(clX) in OpenCL-device
clMFFTD(M,J,N,&clX,&clZ,1);
// perform clG=MFFT(clY) in OpenCL-device
clMFFTD(M,J,N,&clY,&clG,1);
// perform clG=MMul(clZ,clY) in OpenCL-device
clMMulD(M,J,N,&clZ,&clY,&clG);
// perform clZ=iMFFT(clG) in OpenCL-device
clMFFTD(M,J,N,&clG,&clZ,0);
// read the result from clZ to the Host in matrix Z
clEnqueueReadBuffer(cmndQ,clZ,
 CL_TRUE,0,M*J*N*szC,Z,0,0,0);
}//OpenCL_MFSvrtkaE
```

Коэффициенты ускорения, полученные в результате многократных прогонов OpenCL-программы варианта E, применяющей функцию **OpenCLMFSvrtkaE**, представлены в таблицах 9 (для «Intel») и 10 (для «NVidia»).

Таблица 9. Ускорение множественной свёртки OpenCL-программой варианта E на платформе Intel

M×J: N	2×2	5×5	10 ×10	20 ×20	50 ×50	100 ×100	200 ×200
32	0.045	0.268	1.087	3.795	17.264	32.846	30.792
64	0.093	0.543	2.131	6.819	27.266	37.311	26.393
128	0.178	1.176	4.144	12.470	35.177	33.262	28.811
256	0.357	2.217	7.497	21.544	41.577	30.334	29.431
512	0.704	4.131	12.529	32.044	40.671	35.148	---
1024	1.375	8.698	23.303	38.921	35.608	33.206	---
2048	2.301	12.729	33.158	45.598	36.581	---	---
4096	4.976	21.606	44.070	37.425	25.448	---	---
8192	9.727	39.396	51.513	33.221	---	---	---
16384	17.89	56.034	35.912	34.431	---	---	---
32768	29.57	**56.226**	41.890	38.317	---	---	---
65536	41.44	43.630	37.650	---	---	---	---

Таблица 10. Ускорение множественной свёртки OpenCLпрограммой варианта E на платформе NVidia

M×J: N	2×2	5×5	10 ×10	20 ×20	50 ×50	100 ×100	200 ×200
32	0.081	0.462	2.179	6.825	38.500	53.563	70.350
64	0.174	1.030	4.517	14.814	51.103	71.054	80.506
128	0.315	2.484	15.920	39.250	66.743	75.309	104.98
256	0.800	5.450	19.909	49.571	80.315	99.286	86.500
512	1.464	8.822	39.333	73.900	126.02	115.78	114.41
1024	3.478	17.138	55.389	93.588	124.29	125.99	124.38
2048	6.214	38.697	95.664	122.75	162.68	132.78	---
4096	13.07	58.976	107.60	145.44	147.75	143.83	---
8192	25.61	80.769	131.46	163.89	**174.72**	---	---
16384	36.70	106.58	158.36	210.86	206.63	---	---
32768	60.58	143.99	185.47	201.98	---	---	---
65536	98.83	173.78	201.76	**229.01**	---	---	---

Они показывают, что вариант E OpenCL-программы во многих случаях обеспечивает более высокие коэффициенты ускорения. В частности, для набора параметров M×J=50×50 и

N=8192 на платформе «NVidia» коэффициент ускорения варианта **E** достигает значения **174.72**, что больше не только значения 151.05 варианта B, но и значения 154.82 варианта C.

6. Сравнение коэффициентов ускорения OpenCL-программ различных вариантов

Для выбора оптимального способа ускорения операции множественной свёртки с применением технологии OpenCL сравним показатели ускорения различных вариантов, рассмотренных ранее OpenCL-программ. Для этого соберём полученные на каждой платформе («Intel» и «NVidia») показатели ускорения в единые сводные таблицы (см. таблицы 11, 12).

В каждой строке этих таблиц приводятся показатели ускорения операции множественной свёртки, полученные для заданного набора параметров (M, J, N) пятью различными вариантами программ (A, B, C, D, E), разработанных с применением технологии OpenCL. При этом отмечается (жирным шрифтом) тот вариант, который обеспечивает для заданных значений параметров наилучшее ускорение операции.

Таблица 11. Ускорение множественной свёртки OpenCL-программами на платформе Intel

N	M×J	A	B	C	D	E
32	2×2	0.148	0.153	**0.166**	0.043	0.045
	5×5	0.841	0.919	**0.978**	0.266	0.268
	10×10	3.405	3.733	**3.890**	1.077	1.087
	20×20	11.920	**13.372**	12.450	3.978	3.795
	50×50	29.990	37.219	**38.678**	16.483	17.264
	100×100	24.522	39.701	**42.702**	26.564	32.846
	200×200	22.771	40.897	**46.873**	22.720	30.792
64	2×2	0.273	0.282	**0.291**	0.091	0.093
	5×5	1.501	1.586	**1.716**	0.548	0.543
	10×10	5.206	6.002	**6.074**	2.101	2.131
	20×20	16.855	**21.210**	19.584	7.148	6.819
	50×50	26.474	38.644	**36.668**	23.940	27.266
	100×100	22.070	32.144	**41.656**	28.269	37.311
	200×200	20.911	30.269	**45.263**	21.207	26.393
128	2×2	0.360	0.355	**0.388**	0.172	0.178
	5×5	2.315	2.398	**2.457**	1.165	1.176
	10×10	8.192	**8.755**	8.519	4.053	4.144
	20×20	22.874	**26.735**	25.621	12.510	12.470
	50×50	22.561	30.126	**38.743**	28.755	35.177
	100×100	19.963	26.496	**41.033**	27.373	33.262
	200×200	19.905	26.554	**46.240**	23.861	28.811
256	2×2	0.468	0.477	**0.521**	0.354	0.357
	5×5	2.649	2.684	**2.973**	2.201	2.217
	10×10	10.749	11.080	**11.210**	7.887	7.497
	20×20	24.742	**31.281**	29.956	19.826	21.544
	50×50	21.691	27.604	16.262	33.537	**41.577**
	100×100	19.392	23.968	13.731	25.227	**30.334**
	200×200	20.897	28.145	15.549	24.365	**29.431**
512	2×2	0.548	0.549	0.613	0.696	**0.704**
	5×5	2.964	2.943	3.217	4.088	**4.131**
	10×10	10.244	10.547	11.033	**13.143**	12.529
	20×20	13.163	14.664	12.008	29.752	**32.044**
	50×50	4.698	4.757	4.880	32.775	**40.671**
	100×100	4.545	4.706	4.862	29.392	**35.148**
1024	2×2	0.600	0.592	0.659	**1.405**	1.375
	5×5	3.539	3.551	3.709	8.227	**8.698**
	10×10	10.498	10.865	11.132	22.293	**23.303**
	20×20	10.034	10.882	10.132	33.875	**38.921**
	50×50	4.396	4.534	4.572	30.287	**35.608**
	100×100	3.970	4.094	4.241	28.413	**33.206**
2048	2×2	0.613	0.632	0.624	2.204	**2.301**
	5×5	2.855	2.991	3.131	12.200	**12.729**
	10×10	9.664	8.946	9.648	27.474	**33.158**
	20×20	8.801	9.431	8.736	38.430	**45.598**
	50×50	3.904	3.995	4.023	31.317	**36.581**
4096	2×2	0.641	0.615	0.701	**5.027**	4.976
	5×5	3.012	3.049	2.781	16.844	**21.606**
	10×10	9.181	9.558	9.610	37.649	**44.070**
	20×20	7.070	7.894	7.158	32.421	**37.425**
	50×50	3.944	4.025	4.025	**30.034**	25.448
8192	2×2	0.670	0.661	0.656	7.178	**9.727**
	5×5	3.101	3.114	3.244	37.084	**39.396**
	10×10	8.320	8.433	7.718	43.632	**51.513**
	20×20	6.311	6.487	6.642	30.501	**33.221**
16384	2×2	0.645	0.643	0.688	17.383	**17.892**
	5×5	3.229	3.273	3.131	49.275	**56.034**
	10×10	6.590	6.721	6.326	31.167	**35.912**
	20×20	5.992	6.253	6.100	28.455	**34.431**
32768	2×2	0.648	0.649	0.658	28.329	**29.571**
	5×5	2.635	2.661	2.426	48.156	**56.226**
	10×10	5.764	5.930	6.169	37.050	**41.890**
	20×20	5.611	5.961	5.596	34.616	**38.317**
65536	2×2	0.661	0.660	0.680	37.278	**41.438**
	5×5	2.199	2.237	2.156	39.669	**43.630**
	10×10	6.379	6.548	6.065	34.137	**37.650**

Таблица 12. Ускорение множественной свёртки OpenCL-программами на платформе NVidia

N	M×J	A	B	C	D	E
32	2×2	0.116	0.233	**0.400**	0.063	0.081
	5×5	0.787	1.308	**2.576**	0.429	0.462
	10×10	2.881	5.000	**8.095**	1.441	2.179
	20×20	9.414	17.063	**27.300**	5.151	6.825
	50×50	30.250	65.154	**77.000**	18.822	38.500
	100×100	32.962	65.923	**81.619**	30.607	53.563
	200×200	36.077	74.053	**93.800**	37.520	70.350
64	2×2	0.337	0.500	**0.814**	0.164	0.174
	5×5	1.802	2.646	**3.943**	0.954	1.030
	10×10	6.281	11.167	**16.080**	3.865	4.517
	20×20	17.694	35.389	**37.471**	9.800	14.814
	50×50	34.965	71.179	**79.720**	28.884	51.103
	100×100	40.192	75.075	**88.422**	38.260	71.054
	200×200	42.344	77.631	**98.803**	44.361	80.506
128	2×2	0.539	0.710	**1.000**	0.277	0.315
	5×5	3.640	4.440	**7.114**	1.932	2.484
	10×10	15.920	26.534	**31.841**	13.267	15.920
	20×20	31.400	52.333	**52.333**	22.429	39.250
	50×50	42.473	93.440	**116.800**	46.720	66.743
	100×100	48.575	80.958	**107.945**	51.132	75.309
	200×200	49.209	98.419	**121.131**	52.490	104.980
256	2×2	0.727	0.941	**1.311**	0.640	0.800
	5×5	4.955	6.812	**8.385**	4.192	5.450

	10×10	11.526	19.909	**24.333**	14.600	19.909
	20×20	31.545	57.833	**69.400**	38.556	49.571
	50×50	54.953	104.410	**116.011**	49.719	80.315
	100×100	52.125	94.773	**115.833**	54.868	99.286
	200×200	52.175	99.606	**117.393**	57.667	86.500
512	2×2	1.206	1.708	**1.864**	1.367	1.464
	5×5	6.438	8.822	**9.528**	5.955	8.822
	10×10	16.857	24.842	27.765	24.842	**39.333**
	20×20	36.950	67.182	67.182	43.471	**73.900**
	50×50	64.714	104.104	126.021	63.011	**126.021**
	100×100	60.089	107.887	**131.861**	65.027	115.781
	200×200	58.099	104.742	**127.998**	62.493	114.411
1024	2×2	1.429	1.818	1.818	2.963	**3.478**
	5×5	6.716	8.569	8.283	14.200	**17.138**
	10×10	18.811	26.237	27.694	35.607	**55.389**
	20×20	44.194	72.318	75.762	54.862	**93.588**
	50×50	66.867	132.063	**135.449**	74.402	124.294
	100×100	64.929	106.904	**139.257**	65.736	125.994
	200×200	61.224	112.536	**131.292**	64.772	124.382
2048	2×2	1.526	1.723	1.513	4.807	**6.214**
	5×5	7.287	8.132	8.502	28.774	**38.697**
	10×10	24.404	31.888	32.319	56.943	**95.664**
	20×20	47.576	80.052	78.349	68.193	**122.747**
	50×50	77.568	139.437	148.262	85.494	**162.676**
	100×100	69.470	119.994	**140.243**	76.064	132.775
4096	2×2	1.499	1.578	1.596	10.455	**13.069**
	5×5	7.874	8.574	8.825	40.300	**58.976**
	10×10	23.853	29.346	30.263	64.561	**107.602**
	20×20	56.910	83.549	82.670	83.549	**145.437**
	50×50	82.800	137.850	146.565	84.556	**147.751**
	100×100	75.480	129.630	**149.873**	83.597	143.826
8192	2×2	1.454	1.490	1.480	23.472	**25.606**
	5×5	8.210	8.794	8.929	55.851	**80.769**
	10×10	26.761	31.208	31.300	79.674	**131.463**
	20×20	59.489	86.613	84.426	90.359	**163.886**
	50×50	84.016	151.045	154.819	93.440	**174.717**
16384	2×2	1.499	1.517	1.530	28.231	**36.700**
	5×5	8.515	9.042	9.196	68.284	**106.575**
	10×10	26.673	31.106	31.495	90.944	**158.357**
	20×20	77.436	111.783	111.149	121.357	**210.856**
	50×50	100.41	174.666	---	114.121	**206.627**
32768	2×2	1.395	1.416	1.781	44.668	**60.578**
	5×5	8.121	8.488	10.298	78.675	**143.988**
	10×10	24.688	27.812	31.106	105.532	**185.465**
	20×20	65.389	90.039	65.087	113.569	**201.975**
65536	2×2	1.168	1.175	1.205	54.032	**98.826**
	5×5	6.290	6.455	6.635	105.052	**173.778**
	10×10	21.880	23.852	23.705	113.889	**201.758**
	20×20	33.502	38.298	---	131.304	**229.012**

Анализируя эти таблицы, можно заметить, что при N≥512 вариант **E** даёт наилучшие показатели ускорения (как на платформе «Intel», так и на платформе «NVidia») почти на всех наборах параметров (M,J,N). Но при N<512 (и на платформе «Intel», и на платформе «NVidia») для многих наборов параметров вариант E всё же уступает варианту C как наиболее оптимальному. И объясняется это тем, что процедуры ядра варианта C (как и в вариантах A,B) эффективно используют быстродействующую локальную (private) память исполнителей OpenCL-среды для размещения в ней обрабатываемых векторов.

Поскольку при каждом вызове процедуры ядра вариантов D и E совершают лишь одноразовые действия над элементами векторов, расположенных в заданных матрицах в глобальной памяти, то никакой выгоды от сохранения элементов этих векторов в локальной памяти получить не представляется возможным.

Преимущество варианта C как раз и проявляется в тех случаях, когда объём такой локальной памяти позволяет хранить в ней полностью один (XP) или даже два (XP,YP) вектора, над которыми выполняется интенсивная обработка в процедурах ядра. Но это преимущество тут же исчезает, когда текущий обрабатываемый вектор (один YP или оба YP и XP) становится невозможно целиком разместить в этой локальной памяти при больших длинах (N) векторов.

Заметим, что это ограничение, связанное с максимально возможным объёмом локальной памяти исполнителей OpenCL-среды, на разных OpenCL-платформах проявляется при разных значениях N. Так, на платформе «Intel» такое ограничение возникает уже при N=512, а на платформе «NVidia» − лишь при N=8192.

Подобное ограничение приходится как-то учитывать при составлении процедур ядра. Для этого можно, например, предусмотреть компоновку разных вариантов процедур, используя директивы условной компиляции. В одном варианте (при допустимой длине N) обрабатываемые вектора XP, YP располагать в локальной памяти (как это и было продемонстрировано в рассмотренных ранее процедурах). А в другом варианте сразу настраивать указатели векторов XP, YP на те участки глобальной памяти, куда они впоследствии и будут записываться.

Чтобы учесть обозначенное ограничение и предусмотреть различные варианты компоновки OpenCL-программы на любой из рассмотренных платформ, а также обеспечить работоспособность полученной OpenCL-программы при любых значениях длин векторов N, в процедурах ядра вариантов A, B ,C объявления векторов XP и YP были скорректированы так:

```
#if NP<=MAX_NP
 complex XP[NP];
#else
__global complex *XP; XP=VZ;
#endif
#if NP*2<=MAX_NP
 complex YP[NP];
#else
__global complex *YP; YP=VY;
#endif
```

Величины **NP** и **MAX_NP** задают соответственно текущую и максимально возможную (для данной платформы) длину комплексных

векторов, которые процедуры ядра могут разместить в локальной памяти исполнителей. Они определяются как define-переменные при компиляции файла “MFSvertka.cl”, содержащего исходный текст процедур ядра на языке OpenCL. Поясним, как обеспечить требуемую настройку величин **NP** и **MAX_NP** без необходимости корректировать компилируемый файл “MFSvertka.cl” всякий раз для нового значения N длины обрабатываемых векторов, которое может задаваться, например, в качестве параметра командной строки при запуске OpenCL-программы.

Ранее (в [3,п.2.4]) уже было продемонстрировано, как можно в OpenCL-программе осуществить компиляцию процедур ядер, сосредоточенных в файле. Для этого надо предварительно прочитать содержимое этого текстового файла в одну или несколько строк. Затем подключить в OpenCL-контекст новый программный объект **prgrm** и откомпилировать его в универсальный внутренний код, который далее будет приниматься на исполнение драйвером OpenCL-устройства:

```
cl_program prgrm
prgrm=clCreateProgramWithSource
(cntxt,1,(char**)&cSrcStr,0,0);
clBuildProgram(prgrm,0,NULL,NULL,
 BPrgrmOpt,NULL,NULL);
```

При вызове функции **clBuildProgram** можно поставить (5-м параметром) непустой указатель на строку, в которой можно задать опции такой компиляции. А саму эту строку следует предварительно подготовить такой, чтобы в ней были сформированы необходимые определения для define-переменных NP и MAX_NP:

```
char *Opt="-D NP=%d -D MAX_NP=%d";
 sprintf(BPrgrmOpt,Opt,N,MAX_NP);
```

В итоге значение NP принимается равным длине обрабатываемых векторов, т.е. значению N, которое задаётся в самой OpenCL-программе явно или передаётся в качестве параметра командной строки при запуске программы. А значение MAX_NP задаётся в OpenCL-программе с помощью директивы #define в зависимости от того, для какой платформы она компонуется:

```
#if USE_OPENCL==1 /* платформа INTEL
*/
#define MAX_NP 512
#elif USE_OPENCL==2 /* пл. NVIDEA */
#define MAX_NP 16384
#else
#define MAX_NP 256
#endif
```

7. Заключение

Таким образом, технологию OpenCL можно успешно применять для повышения быстродействия таких вычислительных программ, в которых требуется выполнять над множеством векторов, сгруппированных в многомерные массивы данных, хотя и весьма сложные, но, по сути, однотипные вычислительные операции.

Однако для этого требуется построить такую программу, которая сумеет обеспечить в среде OpenCL исполнение этих однотипных операций над разными векторами путём организации одновременной совместной вычислительной работы необходимого количества параллельно функционирующих исполнителей.

Представленные в статье примеры OpenCL-программ показывают, что операцию свёртки для многомерных массивов комплексных векторов в среде OpenCL можно ускорить даже в большей степени, чем множественную операцию быстрого преобразования Фурье для массивов комплексных векторов такого же размера.

Публикация выполнена в рамках государственного задания НИЦ «Курчатовский институт» — НИИСИ по теме «Методы разработки аппаратно-программных платформ на основе защищенных и устойчивых к сбоям систем на кристалле и сопроцессоров искусственного интеллекта и обработки сигналов», шифр № FNEF-2024-0003.

Acceleration of Fast Convolution Operation for Multiple Complex Vectors Based on OpenCL Technology

A.A. Burtsev

Abstract. The article is devoted to the use of OpenCL technology, which allows using the powerful resources of graphic processors to improve the performance of computing programs. Techniques for developing effective parallel programs in the OpenCL environment are considered to speed up the convolution operation for multiple complex vectors based on the fast Fourier transform.

Keywords: parallel programming, OpenCL technology, heterogeneous systems, fast Fourier transform, convolution operation.

Литература

1. В.В. Воеводин, В.В. Воеводин. Параллельные вычисления. Спб., БХВ-Петербург, 2004.

2. Официальный OpenCL–сайт организации Khronos Group, http://www.khronos.org/opencl/

3. А.А. Бурцев. Оптимизация операции перемножения матриц на основе технологии OpenCL. «Труды НИИСИ РАН», Т. 10 (2020), № 5-6, 100–112.

4. А.А. Бурцев. Применение технологии OpenCL для ускорения вычисления интегралов. // «Труды НИИСИ РАН», Т.13 (2023), №1-2, 19-24.

5. А.А. Бурцев. Ускорение быстрого преобразования Фурье на основе технологии OpenCL. «Труды НИИСИ РАН», Т. 11 (2021), № 4, 27–37.

6. А.А. Бурцев. Оптимизация операции быстрого преобразования Фурье в среде OpenCL. // «Труды НИИСИ РАН», Т.12 (2022), №1-2, 11-27.

7. А.А. Бурцев. Ускорение быстрого преобразования Фурье для многомерных массивов комплексных векторов на основе технологии OpenCL. // «Труды НИИСИ РАН», Т.14 (2024), №2, 22-30.

8. Стивен Смит. Цифровая обработка сигналов. Практическое руководство для инженеров и научных работников. М., Додека-XXI, 2012.

Коррекция многопроцессорных расписаний в режиме реального времени

М.Г. Фуругян

ФИЦ ИУ РАН, Москва, Российская Федерация;

rtsccas@yandex.ru

Аннотация. Рассматривается задача планирования вычислений в многопроцессорных системах при периодически поступающих запросах на выполнение комплекса работ с нефиксированными длительностями. Используется неоднородный набор ресурсов – возобновляемых и невозобновляемых. Рассмотрены случаи, когда работы допускают прерывания и переключения с одного процессора на другой, а также когда работы непрерываемые. Разработаны алгоритмы, в которых при обработке каждого запроса корректируется расписание, построенное при обработке предыдущего запроса. Алгоритмы основаны на сетевом моделировании и поиске максимального потока и потока минимальной стоимости.

Ключевые слова: многопроцессорная система, директивный интервал, допустимое расписание, потоковая сеть, максимальный поток, поток минимальной стоимости.

1. Введение

Одно из основных требований, предъявляемых к вычислительным системам реального времени, заключается в том, что все операции должны выполняться в строго отведенных для них временных интервалах. Например, при испытании и функционировании некоторых сложных технических объектов (самолеты, электростанции, ядерные реакторы) запросы на выполнение вычислительных заданий могут поступать с очень высокой частотой и должны успевать обрабатываться в заданном темпе. В случае, когда длительности выполнения всех программных модулей фиксированы, оптимальное расписание их выполнения можно составить заранее, т.е. до проведения эксперимента.

В случае, когда длительности становятся известны в момент поступления запроса на выполнение комплекса вычислительных заданий, расчет расписания производится в режиме реального времени. Это требует эффективных алгоритмов построения оптимальных расписаний. Этим вопросам посвящено большое количество публикаций.

Так, в [1, 2] рассматриваются задачи построения прерываемых расписаний в многопроцессорных системах при заданных директивных интервалах для каждого задания. При выборе подходящего математического аппарата для построения оптимальных расписаний и временных диаграмм выполнения программных модулей в системах реального времени авторы провели сравнительный анализ обобщенных сетей петри и конечных автоматов с остановкой таймера. Предпочтение было отдано последним.

Публикации [3, 4] посвящены построению расписаний выполнения комплекса непрерываемых работ с нефиксированными параметрами (длительностями и ресурсами). Авторами разработан алгоритм, позволяющий разбивать область изменения параметров на многогранники устойчивости, внутри каждого из которых структура оптимального по быстродействию расписания остается неизменной. Способ построения расписаний основан на использовании метода ветвей и границ.

В [5. 6] авторы описывают и проводят сравнительный анализ эвристических и генетических алгоритмов, а также методов с самообучением для построения многопроцессорных расписаний без прерываний.

В [7] исследованы задачи планирования прерываемых работ с директивными интервалами в системах с неоднородным набором ресурсов – возобновляемыми (которые могут использоваться повторно, например, процессоры, приборы, различные механизмы) и невозобновляемыми (которые повторно использоваться не могу, например, электроэнергия, финансы, горюче-смазочные материалы). Разработаны алгоритмы построения оптимальных расписаний, основанные на сведении исходной задачи к нахождению потоков с заданными характеристиками в специальных сетях.

В настоящей статье рассматривается задача построения допустимого многопроцессорного расписания для комплекса работ с директивными интервалами. Рассмотрены случаи, когда работы допускают прерывания и переключения с одного процессора на другой, а также когда работы непрерываемые. Система включает в себя ресурсы двух типов – возобновляемые и невозобновляемые. Предполагается, что запросы на

выполнение комплекса работ поступают периодически, а длительности работ не являются фиксированными и становятся известными только в моменты поступления запросов. Разработаны алгоритмы, которые при поступлении каждого запроса используют расписание, построенное при обработке предыдущего запроса. Это сокращает время построения расписания, что имеет большое значение для обработки информации в реальном масштабе времени. Алгоритмы основаны на поиске потока минимальной стоимости и максимального потока в специальных сетях.

2. Постановка задачи

Имеется комплекс работ (заданий) $W = \{w_1, w_2, \dots, w_n\}$, запросы на выполнение которого поступают в моменты времени $\tau_1, \tau_2, \dots, \tau_k, \dots$ При поступлении запроса τ_k для каждой работы w_i устанавливается директивный интервал ее выполнения $[\tau_k + b_i;\ \tau_k + f_i]$, $f_i > b_i$, $\tau_k + f_i \leq \tau_{k+1}$, $i = \overline{1,n}$, $k = 1, 2, \dots$ Для выполнения работ имеются m идентичных процессоров (возобновляемые ресурсы) и невозобновляемый ресурс. В момент времени τ_k его объем составляет R_k и он распределяется между работами W, а при последующих запросах он использоваться не может. При обработке k-го запроса длительность выполнения работы w_i, если ей не выделен невозобновляемый ресурс, составляет t_i^k, $t_i^k \geq f_i - b_i$, $i = \overline{1,n}$, $k = 1, 2, \dots$. Эта величина становится известной только в момент времени τ_k. Если же невозобновляемый ресурс в объеме r_i^k выделен, то длительность выполнения работы w_i будет составлять $t_i^k - a_i r_i^k$. Здесь $a_i \geq 0$ – известная величина, $t_i^k - a_i r_i^k > 0$,

$$\sum_{i=1}^{n} r_i^k \leq R_k, k = 1, 2, \dots \quad (1)$$

Распределение невозобновляемого ресурса, удовлетворяющее ограничениям (1), называется допустимым.

Рассматриваются случаи, когда (1) при выполнении работ допускаются прерывания и переключения с одного процессора на другой, а соответствующие временные издержки при этом не учитываются; (2) работы являются непрерываемыми. Допустимым расписанием для комплекса работ W называется расписание, при котором каждая работа выполняется в своем директивном интервале. Не допускается параллельное выполнение нескольких работ одним процессором и одновременное выполнение одной работы несколькими процессорами. Допустимым решением будем называть пару “допустимое распределение невозобновляемого ресурса – допустимое расписание комплекса работ W”.

Требуется разработать алгоритм, который для каждого τ_k строит допустимое решение. При этом следует учесть, что длительности работ становятся известными только в момент поступления очередного запроса на их выполнение. Поэтому искомый алгоритм будет работать в режиме реального времени, что накладывает жесткие требования на его быстродействие. Предлагаемый алгоритм основан на том, что при построении решения для очередного запроса используется решение, полученное для предыдущего запроса.

3. Алгоритм решения задачи для случая прерываемых работ

Для решения поставленной задачи в случае прерываемых работ воспользуемся алгоритмом, разработанным в [8]. Для поиска решения S_1 при обработке запроса, поступившего в момент времени τ_1, как описано в [8], построим потоковую сеть G_1, в которой определим поток g_1 минимальной стоимости $c(g_1)$. Для его поиска может быть использован, например, алгоритм дефекта [9]. Как следует из [8], решение S_1 существует в том и только том случае, когда

$$c(g_1) \leq R_1 - \sum_{i=1}^{n} t_i^1 / a_i.$$

В [8] также указано, какой объем невозобновляемого ресурса следует выделить каждой работе w_i и как по найденному потоку построить допустимое расписание. Вычислительная сложность построения решения S_1 составляет

$$O\left(n^4 \sum_{i=1}^{n} t_i^1\right).$$

Для поиска решения S_kпри обработке запроса, поступившего в момент $\tau_k, k \geq 2$, можно воспользоваться решением, построенном при обработке запроса, поступившего в момент τ_{k-1}. А именно, при поиске потока минимальной стоимости g_k в сети G_k с помощью алгоритма дефекта в качестве начального потока можно взять не нулевой поток, а поток минимальной стоимости g_{k-1}, построенный ранее в сети G_{k-1}. В этом случае вычислительная сложность алгоритма будет составлять

$$O\left(n^4 \sum_{i=1}^{n} \left|t_i^k - t_i^{k-1}\right|\right),$$

а не

$$O\left(n^4 \sum_{i=1}^{n} t_i^k\right),$$

если алгоритм стартовал бы с нулевого потока. Такой выигрыш во времени работы алгоритма может иметь существенное значение при

проведении расчетов в реальном времени.

Рассмотрим теперь частный случай, когда для выполнения работ используются только процессоры (невозобновляемый ресурс отсутствует, т.е. $R_k = 0, k = 1, 2, \dots$) и, кроме того, $t_i^k \geq t_i^{k-1}$ при всех $k = 1, 2, \dots, i = \overline{1, n}$. В этом случае для построения решения эффективнее воспользоваться алгоритмом, описанным в [10], который основан на поиске максимального потока в специальной сети Н. А для поиска максимального потока можно воспользоваться алгоритмом кубической сложности, описанным в [11].

Предположим, что при обработке запроса, поступившего в момент τ_{k-1}, была построена сеть H_{k-1} и найдем в ней максимальный поток h_{k-1}. Пусть, далее, при поиске максимального потока было выполнено $p < n$ этапов [11]. Тогда при обработке запроса, поступившего в момент τ_k и построении максимального потока h_k в сети H_k в качестве начального потока следует взять поток h_{k-1}, а не нулевой поток. В этом случае вычислительная сложность алгоритма поиска максимального потока h_k будет составлять $O((n-p)n^2)$, а не $O(n^3)$. Такой выигрыш во времени также может иметь большое значение при проведении вычислений в реальном времени.

4. Случай непрерываемых работ

В этом разделе рассматривается задача на быстродействие в случае, когда работы не допускают прерывания и переключения с одного процессора на другой и, кроме того, невозобновляемый ресурс отсутствует ($R_k = 0, k = 1, 2, \dots$). Как известно [12], в этом случае задача является NP-трудной в сильном смысле.

В [13] описан точный алгоритм решения этой задачи, основанный на методе ветвей и границ. При этом нижние и верхние оценки длины оптимального по быстродействию расписания на различных подмножествах множества всех расписаний, возникающих при ветвлении, вычисляются с помощью рекуррентных соотношений и использования структуры данных в виде двоичной кучи (пирамиды). Это позволило сократить трудоемкость соответствующих процедур с $O(mn)$ и $O(m+n)$ (в случае, когда при вычислении оценок рекуррентные соотношения не используются) до $O(n \log_2 m)$ и $O(1)$ соответственно.

Рассмотрим теперь приближенный “жадный” алгоритм, суть которого заключается в следующем [12]. Фиксируем некоторое k и пусть T_j^k – временная загруженность j-го процессора (т.е. сумма длительностей работ, назначенных на него).

Шаг 1. Положить $T_j^k = 0, j = \overline{1, m}$.

Для каждого $i = \overline{1, n}$ выполнять шаги 2, 3.

Шаг 2. Найти $\min\limits_{j=\overline{1,m}} T_j^k = T_{j_0}^k$.

Шаг 3. Работу w_i назначить на j_0 процессор.

Если величины T_j^k хранить в виде двоичной кучи с минимальным элементом в вершине, то вычислительная сложность алгоритма составит $O(n \log_2 m)$. Если после завершения алгоритма были получены величины $T_j^k, j = \overline{1, m}$, то длина полученного расписания равна $T^k = \max\limits_{j=\overline{1,m}} T_j^k$. Пусть T_{min}^k – это длина оптимального по быстродействию расписания. Известно, что

$$T^k / T_{min}^k \leq 1{,}5.$$

Предположим, что известна некоторая оценка снизу L^k величины T_{min}^k: $L^k \leq T_{min}^k$. Тогда можно предложить следующий приближенный алгоритм, основанный на использовании “жадного” алгоритма.

Шаг 1. Для каждого $j = \overline{1, m}$ назначить на j-й процессор работы таким образом, чтобы выполнялось неравенство $T_j^k \leq L^k$.

Шаг 2. Оставшиеся работы (если таковые имеются) назначить на процессоры с помощью “жадного” алгоритма.

Оценим вычислительную сложность этого алгоритма. Если на шаге 1 были назначены p работ, то вычислительная сложность шага 1 составляет $O(p)$, а шага 2 – $O((n-p)\log_2 m)$. Таким образом, вычислительная сложность предложенного алгоритма составляет $O(p + (n-p)\log_2 m)$. При $m > 2$ эта оценка предпочтительнее, чем $O(n \log_2 m)$.

В качестве L^k можно взять, например, величину $(\sum_{i=1}^n t_i^k)/n$. Тогда вычислительная сложность алгоритма составит $O(n + p + (n-p)\log_2 m)$, а неравенство $n + p + (n-p)\log_2 m < n \log_2 m$ выполняется при $m > 2^{\frac{n}{p}+1}$. Например, \если $p = n/2$, то при $m > 8$.

В случае, когда $t_i^{k-1} \leq t_i^k$ при всех $i = \overline{1, n}$, в качестве L^k можно взять величину $\min\limits_{j=\overline{1,m}} T_j^{k-1}$. Тогда вычислительная сложность алгоритма составит $O(p + (n-p)\log_2 m)$ и, как было отмечено выше, такая оценка предпочтительнее той, которая получается при непосредственном применении “жадного” алгоритма.

4.Заключение

Исследована задача планирования вычислений в многопроцессорных системах при периодически поступающих запросах на выполнение

комплекса работ с нефиксированными длительностями. Используется неоднородный набор ресурсов – возобновляемых и невозобновляемых. Рассмотрены случаи, когда работы допускают прерывания и переключения с одного процессора на другой, а также когда работы непрерываемые. Разработаны алгоритмы, в которых при обработке каждого запроса корректируется расписание, построенное при обработке предыдущего запроса. Алгоритмы основаны на сетевом моделировании и поиске максимального потока и потока минимальной стоимости. Определена вычислительная сложность алгоритмов.

Correcting Multiprocessor Schedules in Real Time

M.G. Furugyan

Abstract. The task of planning computing in multiprocessor systems is considered in periodically incoming requests to perform a set of work with non -fixed durations. A heterogeneous set of resources is used - renewable and non -renewable. There are cases when work allow interruptions and switching from one processor to another, as well as when the work is uninterrupted. Algorithms have been developed in which, when processing each request, the schedule built during the processing of the previous request is adjusted. Algorithms are based on network modeling and searching for the maximum flow and flow of the minimum cost.

Keywords: multiprocessor system, directive interval, feasible schedule, flow network, maximum flow, minimum cost flow.

Литература

1. А.Б. Глонина, В.В. Балашов. О корректности моделирования модульных вычислительных систем реального времени с помощью сетей временных автоматов // Моделирование и анализ информационных систем. (2018), Т. 25, № 2, 174 – 192.

2. А.Б. Глонина. Инструментальная система проверки выполнения ограничений реального времени для конфигураций модульных вычислительных систем // Вестник МГУ. Сер. 15. Вычисл. математика и кибернетика. (2020), № 3, 16 – 29.

3. А.В. Мищенко, П.С. Кошелев. Оптимизация управления работами логистического проекта в условиях неопределенности // Известия РАН. Теория и системы управления. (2021), № 4, 123 – 134.

4. М.А. Горский, А.В. Мищенко, Л.Г. Нестерович, М.А. Халиков. Некоторые модификации целочисленных оптимизационных задач с учетом неопределенности и риска // Известия РАН. Теория и системы управления. (2022), № 5, 106 – 117.

5. В.А. Костенко, А.С. Смирнов. Потоковые алгоритмы планирования вычислений в интегрированной модульной авионике // Изв. РАН. ТиСУ. (2019), № 3, 77 – 86.

6. В.В. Балашов, В.А. Костенко, И.А. Федоренко, Ц. Гао, Ч.М. Сун, Ц. Сун. Алгоритм имитации отжига для построения списочных расписаний с ограничениями на количество межпроцессорных передач данных // Автоматика и телемеханика. (2023), № 8, 138 – 152.

7. Д.А. Кононов, М.Г. Фуругян. Распределение неоднородного комплекса ресурсов при региональном планировании в условиях неопределенности // Труды XV межд. конф. “Управление развитием крупномасштабных систем (MLSD'2022)”. – Москва, ИПУ РАН. – 26 – 28 сент. (2022) 952 – 958.

8. Е.О. Косоруков, М.Г. Фуругян. Некоторые алгоритмы распределения ресурсов в многопроцессорных системах // Вестник МГУ. Сер. 15. Вычисл. математика и кибернетика. (2009), № 4, 34 – 37.

9. Э. Майника. Алгоритмы оптимизации на сетях и графах. М.: Мир, 1981, 325 с.

10. Танаев В.С., Гордон В.С., Шафранский Я.М. Теория расписаний. Одностадийные системы. М.: Наука, 1984, 383 с.

11. Кормен Т., Лейзерсон Ч., Ривест Р., Штайн К. Алгоритмы. Построение и анализ. М.: Вильямс, 2005,1296 с.

12. Гэри М., Джонсон Д. Вычислительные машины и труднорешаемые задачи. М.: Мир, 1982, 416 с.

13. Фуругян М.Г. Некоторые алгоритмы решения минимаксной задачи составления многопроцессорного расписания // Изв. РАН, ТиСУ. (2014), №2, с. 50 - 56.

Автоматизация проверки на плагиат: новый подход к анализу кода в цифровой образовательной платформе Мирера

Д.И. Кадина[1], А.Г. Леонов[2], Н.С. Мартынов[3], К.А. Мащенко[4], Э.А. Орлов[5], А.И. Стрекалова[6]

[1] НИЦ «Курчатовский институт» — НИИСИ, Москва, Россия, kadinadaria@mail.ru;

[2] НИЦ «Курчатовский институт» — НИИСИ, Москва, Россия, МГУ им. М. В. Ломоносова, Москва, Россия, МПГУ, Москва, Россия, Государственный университет управления, Москва, Россия, dr.l@vip.niisi.ru;

[3] НИЦ «Курчатовский институт» — НИИСИ, Москва, Россия, nikolai.martynov@math.msu.ru;

[4] НИЦ «Курчатовский институт» — НИИСИ, Москва, Россия, МГУ им. М. В. Ломоносова, Москва, Россия, Государственный университет управления, Москва, Россия, kirill.mashchenko@niisi.ru;

[5] НИЦ «Курчатовский институт» — НИИСИ, Москва, Россия, eric.al.orlov@gmail.com;

[6] НИЦ «Курчатовский институт» — НИИСИ, Москва, Россия, anastasiia.strekalova@math.msu.ru

Аннотация. Задача выявления плагиата в решениях задач по программированию является высокоприоритетной при разработке цифровых образовательных платформ. Это связано с необходимостью оперативно оценивать уровень заимствования в решениях студентов, чтобы динамически формировать финальную оценку выполнения задания. Методы сравнения решений учащихся, основанные только на текстовом соотнесении их частей без привязки к характерным особенностям используемого языка программирования, зачастую не дают точных и достоверных результатов, так же как и статистические методы, основанные на машинном обучении. В данной работе предложен подход, повышающий качество выявления плагиата при блочно-кусочном сравнении студенческих решений, опирающийся на учёт синтаксических особенностей языков программирования.

Ключевые слова: цифровая образовательная платформа, ЦОП Мирера, антиплагиат, антиплагиат-анализ программного кода, блочное сопоставление.

1. Введение

Плагиат исходного кода (Source Code Plagiarism – SCP) решений обучающихся IT-специальностей с использованием цифровых образовательных платформ (ЦОП) является актуальной проблемой [1; 2]. Статистика убедительно показывает, что от 50% до 79% студентов хотя бы раз прибегали к плагиату решений во время обучения [3; 4]. Использования заимствований только увеличилось после ограничений пандемии COVID-19 [5], а также из-за стремительного развития методов искусственного интеллекта, направленных на генерацию кода для решения задач [6].

Методы решения поставленной задачи можно условно разделить на следующие категории: агрегированное сравнение и структурное сравнение.

Первая категория предполагает независимое от порядка элементов сравнение текстов и использует различные агрегатные методы для выявления фундаментальных характеристик анализируемой части. Часто к этой категории относят различные модели машинного обучения. Например, контекстное или стилистическое [6; 7] сравнение решений подразумевает агрегацию кусков решений в некоторое репрезентативное представление с последующим анализом для обнаружения сходства по конкретной исследуемой характеристике. К той же категории относятся методы, использующие векторные представления частей текста, такие как word2vec [8] и doc2vec [9], а также методы, основанные на подсчёте частот в тексте, такие как BOW [10] и TF-ISF [11]. Несмотря на заявленную эффективность приведенных выше методов, к их недостаткам можно отнести отсутствие чувствительности к структурным перестановкам частей программы и высокую склонность к ложноположительным срабатываниям. Также не учитываются синтаксические особенности языков программирования, ведь в коде одинаковые символы используются для совершенно разных выражений языка программирования. К недостаткам методов, основанных на машинном обучении, можно отнести характерные для них недетерминируемость, неинтерпретируемость и слабую предсказуемость результата, что исключает возможность детального описания сравнения, например, для визуализации в решениях, где именно

было совершено заимствование, чтобы преподаватель мог убедиться в достоверности результата работы алгоритма и исключить ложноположительное срабатывание. К тому же, статистические методы легко обходить с помощью добавления фиктивного кода, так как снижается процент значимой части при значительном добавлении «шума» в текст.

Вторая категория предполагает строгую привязку к синтаксису и взаимосвязям внутри анализируемой части текста. Многие современные инструменты обнаружения SCP измеряют сходство между частями текста в целом, но они также не способны учитывать особенности синтаксиса языка программирования и характерную блочную кодовую структуру, где, как правило, каждый блок кода отвечает за некоторую самостоятельную семантическую единицу, которая при списывании переносится с сохранением структурной целостности. Одним из популярных алгоритмов, относящихся ко второй категории, является алгоритм MOSS (Measure Of Software Similarity) [12]. Его работа основана на анализе структурного сходства через токенизацию и хеширование. Исходный код нормализуется: удаляются пробелы, комментарии, переименованные идентификаторы заменяются на стандартные. Затем код разбивается на последовательности токенов (k-граммы), которые хешируются. Используемый в этом методе алгоритм winnowing выбирает подмножество хешей, представляющих текст. Сходство программ определяется по пересечению найденных хешей. Также популярными структурными алгоритмами сравнения являются JPlag [13] и Sherlock [14]. Нужно отметить, что результаты работы таких алгоритмов значительно варьируются в зависимости от методов оценки [15]. Например, JPlag и Sherlock демонстрируют значительные различия в распределении оценок сходства для одних и тех же данных. Недостатком структурных подходов всё ещё является возможность добавления фиктивного кода в уязвимые места, характерные для конкретного алгоритма. Например, MOSSad [16] – автоматизированный фреймворк, использующий генетическое программирование для внедрения семантически нейтральных изменений, вставляет «пустые» операторы (например, объявление неиспользуемых переменных) в «спорные» с точки зрения SCP позиции, нарушая целостность k-грамм. Это приводит к резкому снижению оценок схожести в MOSS. При этом разработчики алгоритма Sherlock в ответ на подобные уязвимости заявляют, что «Если (студенты) могут заниматься плагиатом и избегать обнаружения Sherlock (или JPlag, или MOSS), то они уже могут программировать на хорошем уровне» [14]. Ещё одним подходом из второй категории является построение графов по заданному тексту. Например, был реализован алгоритм построения семантических графов, который можно адаптировать под синтаксис кода с помощью добавления специфических правил [17]. Также существует алгоритм GPLAG, строящий графы и затем оценивающий их покрытия [18].

Отдельно стоит упомянуть комбинирование различных подходов, которое предлагает использование сразу нескольких инструментов обнаружения SCP и дальнейшее формирование взвешенного результата по группе алгоритмов [19].

В настоящей работе предложен автоматический структурный метод обнаружения SCP в обучающих задачах цифровой образовательной платформы, основанный на поиске наибольших совпадающих по заданной метрике строчных блоков кода в различных языках программирования.

2. Реализация алгоритма

Перед процедурой сравнения студенческих решений на предмет наличия SCP необходимо корректно преобразовывать входные данные.

Решения задач по программированию в базе данных цифровой образовательной платформы зачастую хранятся не одним файлом, а сразу архивом из нескольких файлов, которые могут иметь совершенно различный формат, в связи с чем и возникает необходимость в их слиянии в единый текст. Далее из студенческого тексте решения необходимо изъять части шаблона, предоставленного преподавателем, так как этот текст идентичен и фиксирован в рамках одной задачи у всех обучающихся.

Для выбора алгоритмов обработки текста студенческого решения непосредственно перед процедурой сравнения, необходимо рассмотреть возможные модификации текста пользователями для потенциального обхода алгоритма выявления SCP [20]:

1. Изменение комментариев и отступов;
2. Переименование идентификаторов;
3. Изменения в декларациях элементов (например, объявление дополнительных констант, добавление фиктивных строк кода, а также изменение порядка функций и переменных);
4. Изменение операторов программы на семантические эквиваленты (например, for на while, if на switch).

До процедуры сравнения текст решения очищается от пустых строк и комментариев, специ-

фичных для конкретного языка программирования. Увеличение исходного текста за счет «лишних» строк влияет на вероятностную оценку заимствования, так же, как и незначимые пары скобок. Для решения проблемы переименования имен переменных, применяемых студентами с целью обхода алгоритма SCP, подходит метод универсализации идентификаторов в программном коде с помощью замены различных синтаксических единиц на единый заранее зафиксированный стандарт: имён переменных, названий функций и выражений, характерных для конкретного языка программирования. Так, например, для языка C++ имена переменных при обработке приводятся к единой структуре названий имен с единым началом (V) и числовым окончанием, имена функций заменяются на имена с F, имена объектов начинаются с O, а строки (string) в любом формате преобразуются в имена с S. Третья проблема (изменения в декларации элементов) решается переносом строк, в которых происходит эта декларация, в начало кодового блока. Для исключения возможности влияния на процедуру сравнения изменений операторов программы на семантические эквиваленты достаточно зафиксировать один из взаимозаменяемых операторов и преобразовать текст программы, используя только эти операторы.

Следующим этапом обработки текста решений является этап преобразования с использованием регулярных выражений для ряда выделенных случаев в различных языках программирования. Например, в языках C или C++ префиксный оператор ++n, постфиксный оператор n++ и оператор n+=1 хоть и имеют синтаксические различия, зачастую приводят к одному и тому же ожидаемому результату в рамках студенческих задач. Следовательно, подобные конструкции тоже возможно подвергнуть некоторой универсализации.

В предлагаемом алгоритме сравнения текстов решений на наличие SCP используется блочный метод построчных сравнений, основанный на определенном понятии строки в коде программы. Текст программы во многих языках представляет собой последовательность символов и лишь для программиста визуально разделен на строки. Фактически программу можно «склеить» в единую строку, сохранив семантику. Но такой текст для человека сложен для понимания и редактирования, в отличии от представления программы, когда каждая строка несет некий самостоятельный элемент алгоритма. Тогда такую «склеенную» строку необходимо разбить на фрагменты минимальных синтаксических единиц, которые будут представлять минимальные семантические единицы программы. Например, в языке C++ это возможно сделать по концу оператора, символу «;» или по защищённым конструкциям, таким как if или for, что упростит анализ кода и облегчит последующую визуализацию заимствований. Для этого в процессе разбиения необходимо сохранить взаимно однозначное соответствие между индексами разбитых строк и исходными строками для итоговой разметки списанных строк, а также агрегировать результаты анализа в соответствии с заданной вероятностной метрикой плагиата, что дает возможность видеть результат работы алгоритма отдельно для каждой строки.

Дальнейшая работа по выявлению плагиата состоит в сравнении предобработанного студенческого решения (первый файл) с кандидатом на заимствование (второй файл). Таким кандидатом могут быть эталонное решение преподавателя, что говорит о правильности стиля студенческого решения и(или) более ранее (по времени) решение другого студента, чтобы оценить уровень заимствования. При этом для каждой строки из первого сравниваемого файла осуществляется процедура поиска наиболее подходящих кандидатов среди строк второго файла. Учитывая синтаксические особенности текстов решений задач по программированию, связанные с конкретным языком программирования, осуществляется поиск соответствия между последовательными блоками строк. В частности, когда для i-й строки первого файла находится подходящая j-я строка из второго файла, соответствующая заданной метрике плагиата и превышающая установленный порог, учитывающий специфику задач, составленных преподавателем, необходимо проанализировать их ближайшие строки (i+1-ю из первого файла и j+1-ю из второго файла) на предмет совпадения. Это позволит выявить продолжение совпадения и сформировать целый блок совпадений. Процедура сравнения последующих строк продолжается до тех пор, пока не будет нарушено пороговое условие совпадения для очередной пары строк или не завершится обработка текста. В каждом цикле, при выполнении условия для i+k-ой и j+k-ой строк, найденная пара блоков из k строк первого и второго файла добавляется в качестве кандидатов на соответствие с вычисленной усреднённой вероятностной метрикой $((p_1 + \ldots + p_k) / k)$ по строкам блока. В процессе последовательного формирования совпадающего блока сохраняются все промежуточные блоки, что дает возможность разрешать конфликты в процедуре блочного сопоставления.

Сортировка полученных кандидатов осуществляется с учётом блочного и вероятностного приоритета, характерного для задачи выявления SCP. «Жадность» алгоритма поиска совпадений состоит в нахождении наиболее длинных

совпадающих блоков, которые приоритизируются по усреднённой вероятности для блоков одинаковой длины. Итоговая сортировка кандидатов, представляющих собой пары соотнесённых блоков, осуществляется по следующему принципу: приоритетной является длина кандидатов, а для блоков одинаковой длины сортировка производится по убыванию вероятности плагиата, рассчитанной по заданной метрике.

Процедура разметки строк и сопоставления наиболее совпадающих блоков основывается на алгоритме, который гарантирует сходимость, в том числе в процессе разрешения конфликтов, а также выявляет наиболее длинные блоки плагиата.

Сопоставление найденных блоков из первого файла ко второму осуществляется независимо от порядка, но с учётом приоритета, установленного на этапе предварительной сортировки. Кандидаты добавляются в множество сопоставлений по убыванию после сортировки, начиная с длинных блоков, обладающих наибольшей вероятностной метрикой. При обработке блоков длиной k производится сортировка внутри кандидатов по убыванию усредненной вероятностной метрики строк, описанной выше. В случае отсутствия конфликта в виде пересечений с какимилибо уже занятыми строками, кандидат добавляется в результирующее множество сопоставлений плагиата. Если же возникает конфликт с ранее добавленными строками, то кандидат отбрасывается, и вместо него рассматривается более короткий блок, если таковой имеется дальше в отсортированном множестве кандидатов и не пересекается с уже занятыми строками. На Рисунке 1 изображён пример разрешения конфликта между кандидатами при сравнении двух текстовых файлов.

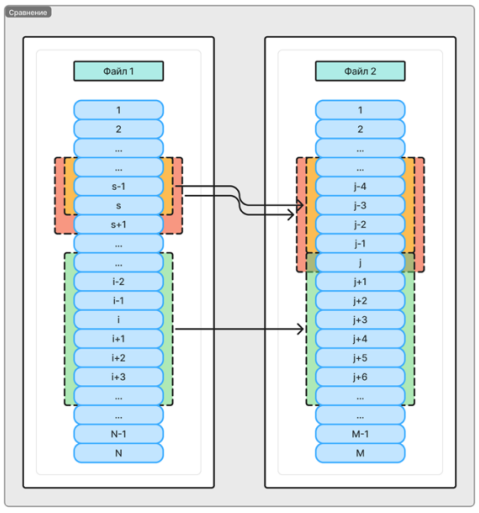


Рис. 1. Пример работы алгоритма разрешения конфликтов между кандидатами

На данном примере в момент, когда очередь доходит до кандидата, выделенного красным цветом, Рисунок 1, в файле 1 соответствующий блок ещё не выбран, однако в файле 2 соответствующие ему строки пересекаются с ранее размеченным зелёным блоком, а именно строка с индексом j содержится сразу в двух этих блоках. В такой ситуации красный блок исключается из рассмотрения, а вместо него рассматривается следующий, меньший по размеру блочный кандидат. Если к моменту, когда очередь дойдёт до распределения жёлтого блока, соответствующие ему строки в файле 2 не будут связаны с кандидатами, находящимися в списке между жёлтым и красным блоком, то жёлтые блоки добавятся к корректному соответствию и зафиксируются в итоговой разметке. Данная итеративная процедура позволяет избегать конфликтов разметки и детерминировать процесс сопоставления, что подчеркивает важность добавления кандидата на каждой итерации построения блока, а не только в момент его формирования с максимальной длиной.

После завершения процедуры блочного сопоставления осуществляется обратный процесс – отображение размеченных строк в исходные, которые, например, изначально могли быть «склеены» в одну, что позволяет агрегировать вероятностные данные для последующего анализа. Данный этап очень важен, чтобы правильно оценить каждый текст и корректно предъявить преподавателю результаты сравнения, поскольку имеет значение вероятность списывания для каждой строки, которая была изначально подана на вход до этапа предобработки. Всё это вместе обеспечивает прозрачность и точность оценки.

Выбор вероятностной метрики, пороги которой существенно влияют на эффективность работы алгоритма, должен осуществляться с учётом специфики выставления баллов за задания и особенностей совпадений между строками в различных языках программирования. Важно корректно определить, какие строки следует считать схожими, а какие не следует. Для большинства популярных языков программирования, использовавшихся в тестировании с реальными пользователями, наилучшей метрикой для оценки и выявления плагиата является расстояние Левенштейна [21]. Эта метрика рассчитывает минимальное количество операций (вставок, удалений и замен), необходимых для преобразования одной строки в другую, что позволяет выявлять не только идентичные фрагменты, но и вариации, возникающие в результате изменений в коде.

3. Заключение

В данной работе рассмотрены существующие подходы и методы по выявлению плагиата исходного кода (SCP) в задачах по программированию. Результатом исследования стало внедрение в цифровую образовательную платформу системы автоматического обнаружения плагиата, которая основана на принципе сопоставления наибольших совпадающих по заданной метрике блоков универсализированного кода, а также учитывающая синтаксические особенности языков программирования и устойчивая к наиболее распространённым способам модификации кода для сокрытия факта списывания.

Разработанный алгоритм был протестирован в рамках эксперимента на реальных учебных данных. Результаты показали высокую степень совпадения оценок вероятности списывания с экспертными ожиданиями преподавателей, а также устойчивость к наиболее распространённым способам модификации кода. Алгоритм продемонстрировал способность выявлять как полные, так и частичные заимствования, при этом сохранял читаемость и интерпретируемость отчётов для преподавателя, что особенно важно для принятия обоснованных решений. Эксперимент подтвердил надёжность и практическую применимость предложенного метода, таким образом, разработанная система представляет собой эффективный инструмент для автоматического выявления плагиата исходного кода в цифровых образовательных платформах.

Работа выполнена в рамках темы государственного задания НИЦ «Курчатовский институт» — НИИСИ по теме № FNEF-2024-0001 (1023032100070-3-1.2.1).

Plagiarism Detection Automation: a New Approach to Code Analysis in the Mirera Digital Educational Platform

D.I. Kadina, A.G. Leonov, N.S. Martynov, K.A. Mashchenko, E.A. Orlov, A.I. Strekalova

Abstract. The task of detecting plagiarism in programming task solutions holds high priority in digital educational platforms due to the necessity of providing accurate and reliable assessment of users' learning progress. Methods that compare submitted solutions solely based on textual similarity, without accounting for the syntactic characteristics of the programming languages in which the solutions are written, often fail to deliver precise and trustworthy results—similarly to statistical approaches based on machine learning. This study proposes a method for block-based comparison of student programming solutions for plagiarism detection, taking into account the syntactic features of programming languages.

Keywords: digital educational platform, DEP Mirera, anti-plagiarism, plagiarism analysis of program code, block-based comparison.

Литература

1. W. Murray. “Cheating in Computer Science”. In: Ubiquity (2010), p. 2. doi: 10.1145/1865907.1865908.

2. G. Cosma and M. Joy. “Towards a Definition of Source-Code Plagiarism”. In: IEEE Transactions on Education (2008), pp. 195–200. doi: 10.1109/te.2007.906776.

3. Curtis, G.J. and Popal, R., 2011. An examination of factors related to plagiarism and a five-year follow-up of plagiarism at an Australian university. International Journal for Educational Integrity, 7(1), pp.30-42.

4. Pierce, J. and Zilles, C., 2017, March. Investigating student plagiarism patterns and correlations to grades. In Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education (pp. 471-476).

5. Maryon, Thomas; Dubre, Vandy C. Mrs.; Elliot, Kimberly; Fagan, Mary Helen; Standridge, Emily; and Lieneck, Christian, "COVID-19 Academic Integrity Violations and Trends: A Rapid Review" (2022). Healthcare Policy, Economics and Management Faculty Publications and Presentations. Paper 1.

6. Ambati, S.H., Stakhanova, N. and Branca, E., 2023, October. Learning AI coding style for software

plagiarism detection. In International Conference on Security and Privacy in Communication Systems (pp. 467-489). Cham: Springer Nature Switzerland.

7. Hourrane, O., 2019. Rich style embedding for intrinsic plagiarism detection. International Journal of Advanced Computer Science and Applications, 10(11).

8. Mikolov, T., Sutskever, I., Chen, K., Corrado, G., Dean, J.: Distributed Representations of Words and Phrases and their Compositionality. arXiv:1310.4546 [cs, stat] (Oct 2013)

9. Le, Q., Mikolov, T.: Distributed representations of sentences and documents. In: Proceedings 31st International Conference on Machine Learning. vol. 32, pp. 1188– 1196 (2014)

10. Zhang, Y., Jin, R. and Zhou, Z.H., 2010. Understanding bag-of-words model: a statistical framework. International journal of machine learning and cybernetics, 1, pp.43-52.

11. El-Rashidy, M.A., Mohamed, R.G., El-Fishawy, N.A. and Shouman, M.A., 2022. Reliable plagiarism detection system based on deep learning approaches. Neural Computing and Applications, 34(21), pp.18837-18858.

12. Schleimer, S., Wilkerson, D.S. and Aiken, A., 2003, June. Winnowing: local algorithms for document fingerprinting. In Proceedings of the 2003 ACM SIGMOD international conference on Management of data (pp. 76-85).

13. Prechelt, L., Malpohl, G. and Philippsen, M., 2002. Finding plagiarisms among a set of programs with JPlag. J. Univers. Comput. Sci., 8(11), p.1016.

14. Joy, M. and Luck, M., 2002. Plagiarism in programming assignments. IEEE Transactions on education, 42(2), pp.129-133.

15. Ahadi, A. and Mathieson, L. (2019). A comparison of three popular source code similarity tools for detecting student plagiarism. In: Proceedings of the Twenty-First Australasian Computing Education Conference, ACE'19, 112–117.

16. Devore-McDonald, B. and Berger, E.D., 2020. Mossad: Defeating software plagiarism detection. Proceedings of the ACM on Programming Languages, 4(OOPSLA), pp.1-28.

17. Леонов А.Г., Мартынов Н.С., Мащенко К.А., Холькина А.А., Шляхов А.В. Автоматизация проверки семантической составляющей текстовых ответов обучающихся в цифровой образовательной платформе // Программные продукты и системы. 2024. Т. 37. № 3. С. 440–452. doi: 10.15827/0236-235X.142.440-452

18. Liu, C., Chen, C., Han, J. and Yu, P.S., 2006, August. GPLAG: detection of software plagiarism by program dependence graph analysis. In Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 872-881).

19. Hayden Cheers, Yuqing Lin, Weigen Yan, Identifying Plagiarised Programming Assignments with Detection Tool Consensus, Informatics in Education 22(2023), no. 1, 1-19, DOI 10.15388/infedu.2023.05

20. Cheers, H., Lin, Y. and Smith, S.P., 2021. Academic source code plagiarism detection by measuring program behavioral similarity. IEEE Access, 9, pp.50391-50412.

21. Levenshtein, V.I., 1966, February. Binary codes capable of correcting deletions, insertions, and reversals. In Soviet physics doklady (Vol. 10, No. 8, pp. 707-710).

Гибридная модель обучения: новации в учете посещаемости в цифровой образовательной платформе Мирера

Н.В. Гриднев[1], А.С. Караваева[2], А.Г. Леонов[3], К.А. Мащенко[4], К.К. Пчелин[5], Е.Д. Тарасюк[6]

[1] НИЦ «Курчатовский институт» — НИИСИ, Москва, Россия, nikita.gridnev@math.msu.ru;

[2] НИЦ «Курчатовский институт» — НИИСИ, Москва, Россия, aleksandrakaravaevaa@yandex.ru;

[3] НИЦ «Курчатовский институт» — НИИСИ, Москва, Россия, МГУ им. М. В. Ломоносова, Москва, Россия, МПГУ, Москва, Россия, Государственный университет управления, Москва, Россия, dr.l@vip.niisi.ru;

[4] НИЦ «Курчатовский институт» — НИИСИ, Москва, Россия, МГУ им. М. В. Ломоносова, Москва, Россия, Государственный университет управления, Москва, Россия, kirill.mashchenko@niisi.ru;

[5] НИЦ «Курчатовский институт» — НИИСИ, Москва, Россия, pcelinkosta0@gmail.com;

[6] НИЦ «Курчатовский институт» — НИИСИ, Москва, Россия, ekaterina.tarasuk@math.msu.ru;

Аннотация. Рассматривается задача автоматизации учета посещаемости студентов в цифровой образовательной платформе Мирера. Разработаны и исследованы два инновационных подхода: временное подтверждение присутствия через QR-коды с применением технологии Server-Sent Events и JWT-токенов, а также введение контрольной метрики методом видеоидентификация студентов с помощью трехмерной реконструкции аудитории. Экспериментально доказано, что предложенные решения способствуют повышению точности и оперативности учета посещаемости, обеспечивая надежный контроль над учебным процессом в условиях смешанной формы обучения.

Ключевые слова: цифровая образовательная платформа, ЦОП Мирера, QR-код, гибридная посещаемость, JWT, Server-Sent Events, WebSocket, модульная архитектура, видеоидентификация, 3D-анализ.

1. Введение

Отечественная цифровая образовательная платформа (ЦОП) Мирера [1] ориентирована на поддержку разнообразных форматов обучения, среди которых выделяются очные, дистанционные и гибридные, то есть смешанные формы. Для поддержки функций мониторинга и сопровождения учебного процесса разработана многоуровневая система учета посещаемости, включающая широкий спектр механизмов подтверждения участия студентов в учебных событиях. Среди используемых методов:

- регистрация стандартных входов в учебные контесты, путем идентификации и аутентификации;
- фиксация нахождения студента в определенном месте посредством предоставления своей геопозиции;
- мониторинг участия в видеоконференциях;
- учет просмотров видеозаписей лекций и занятий;
- отслеживание ознакомления с дополнительными материалами, размещенных преподавателем.

Опыт эксплуатации указанных механизмов подтвердил необходимость создания нового инструмента, направленного на повышение точности и оперативности учета временного и(или) сессионного присутствия студентов. Наибольшую актуальность эта задача приобретает при проведении крупномасштабных аудиторных мероприятий, таких как массовые лекции и семинары, поскольку стандартные методы являются сложновыполнимыми и времязатратными [2].

Для решения вопросов идентификации присутствующих на аудиторных мероприятиях в мире наиболее распространены практики, включающие использование биометрических систем, технологий бесконтактной идентификации (NFC, RFID), мобильного позиционирования (GPS, Wi-Fi) и видеофиксации. Каждый из этих методов заслуживает отдельного рассмотрения.

Биометрическая методика предоставляет высокую точность и минимизацию ложной идентификации, однако сопряжена с высокими исходными инвестициями в оборудование и вызывает беспокойство у участников относительно приватности их персональных данных [3]. NFC

и RFID, будучи также надежными технологиями, обладают существенным недостатком: необходимость идентификационных постов перед входом в аудитории и использование специальных карточек или бейджей студентами, что также увеличивает начальные и эксплуатационные затраты образовательных организаций и, самое главное, не гарантирует совпадение личностей слушателя с носимой карточкой, то есть не исключает манипуляции идентификационными данными [4].

Широко распространенные подходы, основанные на мобильных приложениях, предполагают работу через смартфоны, используя сигналы GPS или Bluetooth. Несмотря на кажущуюся простоту и низкую цену внедрения, такие системы подвержены серьезным проблемам точности, особенно в городских условиях, где сигнал легко искажается высотными зданиями или плохой связью [5].

Отдельную категорию составляют видеосистемы наблюдения, предназначенные для автоматического подсчета посетителей. Подобные решения популярны в определенных областях, но их основным минусом остается низкая эффективность при высокой концентрации обучающихся, при которой наблюдается существенное увеличение ошибочных срабатываний, что снижает достоверность получаемых результатов [6].

При гибридной форме обучения студенты могут находиться как в аудитории с лектором, так и использовать видео-коммуникационные средства связи для участия в занятии, находясь при этом далеко от преподавателя. Эта форма образовательного процесса требует учета и студентов в аудитории, и слушателей, участвующих в занятии через сеть Интернет. Такой метод учета присутствия в настоящей статье называют гибридной посещаемостью.

В данной статье предложены подходы к учету гибридной посещаемости с использованием метода очной посещаемости с помощью динамической QR-верификации с ограниченным временем жизни, а также методики проверки корректности срабатывания способов фиксации очной посещаемости по видеопотоку с помощью адаптированной архитектуры FruitNeRF. Такой подход позволит преодолеть ключевые недостатки традиционных систем учета посещаемости – низкую точность, зависимость от субъективных факторов и недостаточную организацию контроля в условиях смешанного формата обучения, существенно повышая уровень доверия к собираемым данным и скорость их получения.

2. Архитектура и принципы работы системы

2.1. Система учета посещаемости: архитектура и алгоритмы обработки

Многоуровневая система учета посещаемости в ЦОП Мирера реализует комплексный подход к верификации учебной активности, основанный на принципах композитного анализа событий с применением адаптивной логики начисления баллов [7]. Алгоритмы работы системы построены вокруг двух ключевых компонентов: модуля расчета весовых коэффициентов и механизма определения факта посещения.

Система реализована по принципу оптимального начисления баллов при обработке множественных отметок посещаемости. В алгоритме обработки полученных баллов за посещаемость, для каждого типа активности рассчитывается индивидуальный весовой коэффициент по формуле:

$$Weigh = isFulfilled \;\&\&\; isUsed \;?\; typeWeight \;:\; 0,$$

где: isFulfilled – факт выполнения условия, isUsed – активация типа посещаемости в настройках контеста (семинара, занятия, в терминах ЦОП Мирера), typeWeight – заданный вес типа. Особенностью алгоритма является применение функции максимума ко всем рассчитанным весам, что гарантирует студенту учет наиболее выгодного варианта посещаемости (рис. 1). Такая архитектура исключает проблему дублирования баллов при множественной отметке.

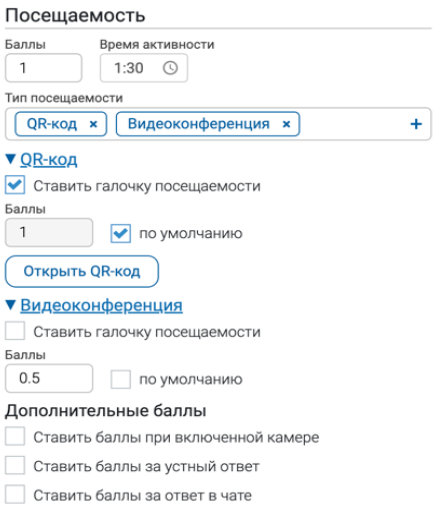


Рис. 1. Пример настроек посещаемости

Система учета посещаемости использует гибкий подход к подтверждению присутствия студентов в зависимости от типа учебной активности. Разные способы проверки классифицируются в зависимости от того, насколько строго они привязаны ко времени проведения занятия.

К первой категории относятся методы с жесткой временной привязкой, действующие исключительно в период активности контеста. К ним относятся: процедура входа в учебный контест, верификация местоположения через геопозицию, сканирование QR-кода, а также участие в видеоконференции занятия. Данные методы требуют синхронного взаимодействия студента с учебным процессом в реальном времени.

Вторая категория включает методы с расширенным временным окном, остающиеся доступными в течение периода, отведенного для досдачи учебных материалов. Примером служит просмотр предоставленных преподавателем видеоматериалов, который может быть засчитан как форма участия даже после завершения основного времени проведения занятия.

К отдельной группе относится пост-активный метод верификации, который сохраняет свою значимость после полного завершения контеста, например, просмотр записи проведенной видеоконференции. Система учитывает его как альтернативную форму учебной активности вне зависимости от реальных сроков проведения исходного занятия. Кроме того, все временные метки и данные о подтверждении присутствия сохраняются в системе независимо от текущих настроек посещаемости. Это гарантирует корректность учета, так как, даже если преподаватель позднее изменит параметры, система автоматически пересчитает все баллы и статусы посещения на основе сохраненных данных.

Для временно-зависимых типов участия, таких как присутствие на видеоконференции, просмотр записей или учебных материалов, реализована градационная система оценивания. В данном случае итоговый балл рассчитывается пропорционально фактическому времени, затраченному студентом на учебную активность, по отношению к установленному нормативному показателю. Система гарантирует, что начисленные баллы не превысят максимально возможное значение для данного типа активности, даже в случае значительного превышения студентом минимальных требований по времени участия.

На примере (рис. 2) показано, что студент присутствовал на занятии очно, и с помощью отправленной геопозиции и присутствии в видеоконференции получил отметку о посещении, однако учитывались только максимальный балл за геопозицию и дополнительные баллы за ответ в чате видеоконференции.

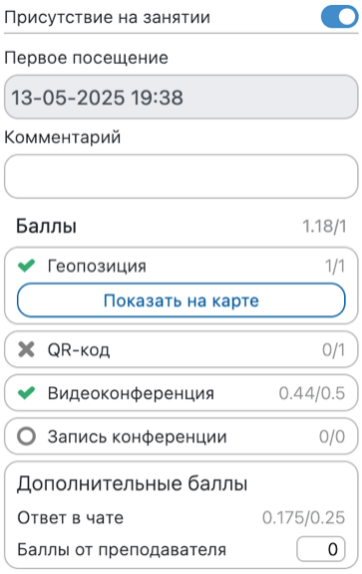


Рис. 2. Пример вычисления баллов и отметки посещаемости

В статистике для преподавателя отображаются возможные и полученные студентом баллы. Типы посещаемости отмечаются следующим образом: зелёная галочка – присутствие, крестик – отсутствие отметки о посещении, кружок – за эту активность нельзя получить посещаемость. Таким образом, обеспечивается не только справедливая оценка как полного, так и частичного выполнения условий посещаемости, но и прозрачная для преподавателя система засчитывания баллов.

2.2. Генезис и обоснование выбора системы QR-верификации посещаемости

Идея внедрения QR-кодов возникла в результате комплексного анализа проблем, выявленных при эксплуатации существующих методов учета посещаемости. В ЦОП Мирера использовались различные подходы к фиксации учебной активности, каждый из которых демонстрировал определенные ограничения в условиях массового обучения [7]. Традиционные способы, такие как учет входа в контест или фиксация геопозиции, хотя и обеспечивали базовый функционал, не удовлетворяли растущим требованиям к надежности, скорости обработки и защите от ложной идентификации.

Первоначальная схема, основанная на автоматической фиксации входа [7] в учебный контест, страдала от фундаментального недостатка – невозможности отличить реальное присутствие студента на занятии от формального доступа к учебным материалам. Геопозиционный

метод, хотя и решал часть проблем аутентификации, создавал новые сложности. Корректность и точность GPS-навигации в городских условиях, особенно внутри университетских корпусов, часто оказывались недостаточными, из-за чего присутствующие студенты не могли отметиться на очных занятиях, однако находящиеся недалеко от аудитории студенты, но не непосредственно на занятии, могли отметиться на занятии. Это вынуждало преподавателей вручную контролировать посещаемость, осуществляя ручной пересчет. При этом гибридный формат обучения, где часть студентов присутствует очно, а часть подключается дистанционно, усложнял задачу. При дистанционном обучении учет входа в видеоконференцию позволял эффективно подтверждать присутствие участников, обеспечивая визуальный контроль. Однако для очных занятий этот метод не подходил.

QR-верификация стала альтернативой для очного формата, устранив проблему неточности геопозиции. С её помощью студенты могли быстро и надежно подтверждать свое присутствие на занятиях без зависимости от качества GPS-сигнала. Преимуществом оказалась возможность строгой временной привязки – короткий срок жизни кода исключал возможность его повторного использования или передачи. Динамическая генерация с частотой 2 секунды обеспечивала необходимый баланс между удобством сканирования и безопасностью от предоставления ложных данных слушателями.

2.3. Реализация посещаемости через QR-код

Механизм учета посещаемости через QR-код представляет собой комплексное решение, сочетающее современные веб-технологии с требованиями образовательного процесса. Система функционирует на основе технологии Server-Sent Events (SSE) (рис. 3), которая была выбрана как оптимальное решение для данного сценария использования.

Принцип работы системы строится вокруг динамически генерируемых QR-кодов с коротким временем жизни (4 секунды) и высокой частотой обновления (каждые 2 секунды). Студенты могут сканировать код двумя способами: либо через интерфейс платформы после входа в систему, либо непосредственно через камеру мобильного устройства – в этом случае система автоматически перенаправляет пользователя на соответствующий контест.

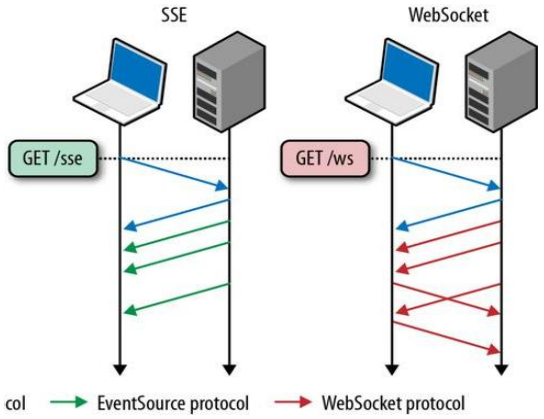


Рис. 3. Схема работы SSE и WebSocket

Выбор SSE в качестве базовой технологии обусловлен несколькими ключевыми факторами. В отличие от WebSocket [8], который предполагает двустороннюю коммуникацию, SSE оптимизированы для сценариев, где требуется преимущественно односторонняя передача данных от сервера к клиенту. Это полностью соответствует задаче регулярной генерации новых QR-кодов. Дополнительными преимуществами являются: более простая реализация, меньшая нагрузка на сервер при массовых подключениях, встроенные механизмы восстановления соединения и естественная совместимость с HTTP/2 [9].

Каждый формируемый QR-код содержит уникальный JWT-токен (JSON Web Token), предназначенный для предотвращения несанкционированного доступа и манипуляций с системой. Токен включает в себя идентификатор конкретного учебного мероприятия и характеризуется коротким периодом своего действия, что исключает возможность его повторной активации или воспроизведения [10]. Реализация серверной составляющей осуществляется с применением механизма активного поддержания соединения, что позволяет оперативно контролировать процессы генерации и обновления QR-кодов для множественных параллельных сессий.

В результате был разработан инновационный алгоритм временной верификации посредством QR-кодирования, основанный на технологии передачи событий от сервера и криптографической защите токенов JSON Web Token. Предлагаемый подход характеризуется простотой технической реализации, незначительной ресурсоемкостью вычислительных мощностей серверной инфраструктуры и высоким потенциалом масштабируемости применительно к мероприятиям массового формата.

Архитектура серверной части оптимизирована для обеспечения непрерывности доставки QR-кодов в режиме реального времени с минимальными задержками, обеспечивая надёжность

аутентификации обучающихся и предотвращение возможных манипуляций. Комбинация технологии SSE с JWT-токенизацией создает надежный механизм, который можно использовать в образовательных платформах, обеспечивая необходимый уровень безопасности при сохранении функциональности в условиях смешанного обучения.

3. Автоматизированная идентификация студентов на основе видеоданных

3.1. Необходимость автоматической видеоиндентификации студентов

Для гибридной формы обучения ЦОП Мирера использует комбинацию методов учета посещаемости, включая QR-верификацию и геопозиционирование. Однако даже при высокой надежности этих методов сохраняется риск технических сбоев и(или) преднамеренных манипуляций. Для минимизации таких рисков введена дополнительная контрольная метрика, которая базируется на автоматизированном анализе видеоданных. При помощи фото-видеофиксации выявляется расхождение данных о посещаемости, полученных через основные методы, и выявлении реальным присутствием студентов в аудитории, что позволяет исключить случаи некорректного засчитывания посещаемости у слушателей и является гарантией для решения спорных вопросов о прогулах в администрации вуза.

Первоначально были использованы методы, основанные на фото данных [7]. При этом на фотографии в больших аудиториях студенты порой могут закрывать друг друга, это является одним из главных аргументов к переходу на идентификацию студентов с помощью видео.

В качестве автоматической системы идентификации студентов на основе видеоданных был выбран подход 3D-реконструкции аудитории и последующая кластеризация в этом пространстве. Такой подход позволяет моделировать расположение студентов и устранять перекрытия для повышения точности идентификации.

3.2. Методология автоматической видеоиндентификация студентов

Для создания 3D-сцены аудитории и подсчета студентов в ней после проведения исследований было принято решение использовать подход FruitNeRF [11] (рис. 4).

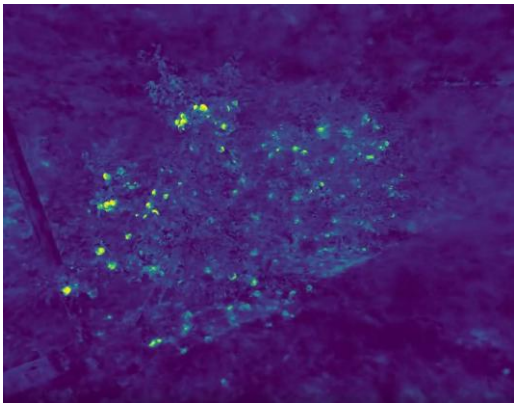


Рис. 4 Подсчет фруктов с помощью FruitNeRF

Эта технология изначально была разработана для подсчета фруктов в сельском хозяйстве. Данный фреймворк, адаптированный под образовательный контекст, использует видео, записанное преподавателем во время обхода аудитории, которое нарезается на множество кадров, имитирующих съёмку с разных ракурсов, что заменяет использование стационарных камер. На основе этих кадров нейросетевая архитектура, основанная на оригинальной архитектуре FruitNeRF, воссоздаёт единую 3D-сцену помещения, анализируя пространственную структуру, текстуры и освещение через комбинацию методов Neural Radiance Fields (NeRF) [12] и семантической сегментации. Алгоритм последовательно обрабатывает кадры, восстанавливает глубину и геометрию пространства, а затем применяет кластеризацию для выделения студентов как отдельных объектов даже если они частично перекрыты на одном из кадров.

Итогом работы становится статическая 3D-модель, на которой автоматически подсчитывается количество студентов, а их позиции фиксируются для последующего анализа посещаемости, сохраняя преимущества FruitNeRF в части точности подсчёта объектов, что решает основные проблемы предыдущего подхода.

3.3. Демонстрация работы автоматической системы видеоидентификации на основе FruitNeRF

Работа системы видеоидентификации студентов на базе адаптированного FruitNeRF включает несколько ключевых этапов, иллюстрированных на примере реальной аудитории:

1. Сбор данных. Преподаватель записывает видео, перемещаясь по аудитории. Видео нарезается на кадры, имитируя съёмку с виртуальных ракурсов, обеспечивая покрытие всего пространства.

2. Сегментация и 3D-реконструкция. Кадры

обрабатываются нейросетью FruitNeRF, которая объединяет методы Neural Radiance Fields (NeRF) и семантической сегментации. Модель восстанавливает 3D-сцену аудитории, учитывая геометрию помещения, текстуры и освещение. Семантическая сегментация отделяет студентов от фона, парт и других объектов.

3. Экспорт облака точек. На основе полей плотности и семантики FruitNeRF генерирует облако точек, соответствующее студентам. Точки фильтруются по порогу плотности, чтобы исключить шум.

4. Кластеризация. Применяется двухэтапная кластеризация: DBSCAN для грубого разделения на кластеры [13]. Алгоритм выделяет зоны высокой плотности, игнорируя шум (рис. 5), а агломеративный подход необходим для уточнения кластеров. Кластеры с перекрытием разбиваются на подгруппы с использованием шаблона студента. Расстояние Хаусдорфа между шаблоном и точками определяет точное количество объектов в кластере.

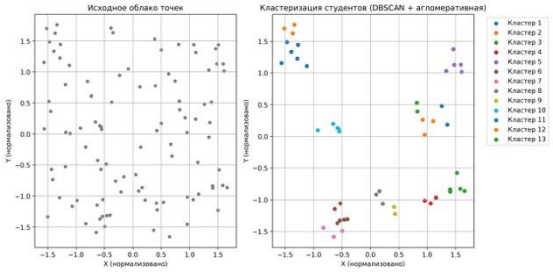


Рис. 5 Визуализация работы алгоритма кластеризации

5. После работы алгоритма преподаватель получает количество студентов в аудитории. Также система автоматически создает 3D-карту сцены (рис. 6), которая сохраняется в ЦОП Мирера.

Рис. 6 3D-карта аудитории, полученная с помощью NeRF методов без постобработки

4. Заключение

Предлагаемые решения демонстрируют существенные достижения в области автоматизации учета посещаемости студентов на цифровой образовательной платформе Мирера. Использование QR-кодов с коротким сроком действия, построенных на основе технологии Server-Sent Events и JWT-токенов, обеспечивает быстрый и безопасный способ подтверждения присутствия студентов.

Предложенный в качестве дополнительной контрольной метрики метод видеоидентификации, созданный на основе адаптированного под образовательный контекст фреймворка FruitNeRF, позволяет точно определять количество присутствующих студентов, даже при их частичном перекрытии, устраняя недостатки традиционных систем видеонаблюдения.

Применение предлагаемых методов создает надежную систему контроля, адаптированную к требованиям современных гибридных форматов обучения.

Работа выполнена в рамках темы государственного задания НИЦ «Курчатовский институт» — НИИСИ по теме № FNEF-2024-0001 (1023032100070-3-1.2.1).

Hybrid Learning Model: Innovations in Attendance Tracking on the Mirera Digital Educational Platform

N.V. Gridnev, A.S. Karavaeva, A.G. Leonov, K.A. Mashchenko, K.K. Pchelin, E.D. Tarasuk

Abstract. The paper addresses the task of automating student attendance tracking within the Mirera digital educational platform. Two innovative approaches have been developed and investigated: time-based presence confirmation using QR codes with Server-Sent Events and JWT tokens, and the introduction of a control metric through student video identification based on 3D classroom reconstruction. Experimental

results demonstrate that the proposed solutions enhance the accuracy and efficiency of attendance tracking, providing reliable control over the educational process in a hybrid learning environment.

Keywords: digital educational platform, DEP Mirera, QR code, hybrid attendance, JWT, Server-Sent Events, WebSocket, modular architecture, video identification, 3D analysis.

Литература

1. Платформа Мирера [Электронный ресурс] URL: https://www.mirera.ru (дата обращения: 05.05.2025)

2. Mohammed K., Tolba A.S., Elmogy M. Multimodal Student Attendance Management System (MSAMS) // Journal of King Saud University - Computer and Information Sciences. — 2020. — Vol. 32, No 1. — P. 99–110. — DOI: 10.1016/j.jksuci.2018.10.004

3. Alzahrani B., Alsolami F. Biometric System: Security Challenges and Solutions // 16th International Conference on Information Technology-New Generations (ITNG 2019). — 2019. — P. 111-117. — DOI: 10.1007/978-3-030-14070-0_17

4. Karsen M., Kurniawan Y., Cassandra C., Juwitasary H. NFC Design for Attendance System in the University // International Journal of Mechanical Engineering and Technology. — 2018. — Vol. 9, No 6. P. 566-571.

5. Chiang T.-W., Yang C.-Y., Chiou G.-J., Lin F.Y.-S., Lin Y.-N., Shen V.R.L., Juang T.T.-Y., Lin C.-Y. Development and Evaluation of an Attendance Tracking System Using Smartphones with GPS and NFC // Applied Artificial Intelligence. 2022, Vol. 36, No 1., P. 1–21. DOI: 10.1080/08839514.2022.2083796

6. Shebbaz M., Ahmed S.M., Alam Z. Computer Vision based Attendance Management System [Электронный ресурс] // YMER Digital. 2024. Vol. 23, No 4. P. 773-778. URL: https://ymerdigital.com/uploads/YMER230420.pdf (дата обращения: 12.07.2024)

7. Леонов А.Г., Мащенко К.А., Шляхов А.В., Холькина А.А. Подходы к учету посещаемости студентов в цифровой образовательной платформе Мирера, Труды НИИСИ РАН, Том 12, № 3

8. Шестаков В.С., Сагидуллин А.С. Применение технологии websocket в web-приложениях технологического назначения // Известия высших учебных заведений. Приборостроение. 2015. URL: https://cyberleninka.ru/article/n/primenenie-tehnologii-websocket-v-web-prilozheniyah-tehnologicheskogo-naznacheniya

9. Hickson I. Server-Sent Events [Электронный ресурс]. W3C Recommendation, 2015. URL: https://www.w3.org/TR/eventsource/ (дата обращения: 05.05.2025)

10. Jones M., Bradley J., Sakimura N. RFC 7519: JSON Web Token (JWT) [Электронный ресурс]. Internet Engineering Task Force (IETF), 2015. URL: https://tools.ietf.org/html/rfc7519 (дата обращения: 05.05.2025)

11. Smith J., Doe R., Chen L. FruitNeRF: A Unified Neural Radiance Field based Fruit Counting Framework // arXiv preprint arXiv:2408.06190. 2024. URL: https://arxiv.org/abs/2408.06190

12.Mildenhall B., Srinivasan P.P., Tancik M., Barron J.T., Ramamoorthi R., Ng R. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis // arXiv preprint arXiv:2003.08934. 2020. URL: https://arxiv.org/abs/2003.08934

13. Ester M., Kriegel H.-P., Sander J., Xu X. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise // Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD'96). Portland, OR, USA. 1996. P. 226–231. URL: https://dl.acm.org/doi/10.5555/3001460.3001507

Modeling the Defense of Weak Prey Agents Against a Strong Predator Agent

V.G. Red'ko

NRC "Kurchatov Institute" — SRISA, Moscow, Russian Federation, vgredko@gmail.com

Abstract. We constructed and studied a model of interaction between a community of relatively weak prey agents and a strong predator agent in a two-dimensional grid world (a lattice environment typical of grid automata and agent-based models).

The predator can attack, kill, and consume prey agents. Each prey agent is controlled by a neural network and adopts one of two behavioral strategies: (1) normal activity, or (2) defense against the predator.

In the normal activity strategy, prey agents lie dormant, feed, breed, and move through the grid. In the defense strategy, they attempt to escape, threaten, or attack the predator. The neural network outputs control each agent's actions. The predator follows a simpler, rule-based protocol: it can lie dormant, evade threatening prey, or attack them. Its behavior is governed by basic logic.

We analyzed the model using computer simulations. We found that, with realistic parameters, the prey agents collectively overcome the predator: prey resource levels increase steadily, while the predator's resources decline to zero, leading to its extinction. We also discovered that successful defense requires a sufficiently abundant food supply; when prey food is scarce, the predator successfully suppresses the prey population. We used computer simulation to analyze the model. When the prey agents' food supply is low, the predator agent suppresses the prey agents.

Keywords: prey agents, predatory agent, prey-predator struggle.

1. Introduction

Models of interaction between autonomous agents have been studied since the early 1990s [1, 2]. For example, L.S. Yaeger [3] and D. Ackley et al. [4] studied populations of competing agents. M. Burtsev et al. [5] researched a rather complex model of evolutionary self-organization and speciation in a population of agents. In some cases, a group of relatively weak agents fights against a stronger agent. It is similar to the attack of a large flock of starlings on a sparrowhawk described by K. Lorenz [6]. V.G. Red'ko et al. [7] created and studied a computer simulation model of interaction between two groups of autonomous agents competing for the territory. It was demonstrated that a successful attack on the agents from an alien group leads to an expansion of the territory occupied by the attacking group. This paper considers a model of interaction between a sufficiently large group of relatively weak prey agents and a strong predator agent in a grid world.

2. Model Overview

Suppose that there is a society of relatively weak prey agents in the grid world. There is also a strong predatory agent. The predator agent can attack, kill, and eat the prey agents. The embedded control system of the prey agent is a simple neural network.

The predator agent has no embedded neural network control system. Its behavior is governed by simple logic presented below.

The world is a 1D chain of cells with the number of cells limited to N. The world is closed: if we move to the right beyond the N^{th} cell, we get to the 1^{st} cell; if we move to the left beyond the 1^{st} cell, we get to the N^{th} cell. Each cell may contain more than one agent.

The cells also have food for the prey agents. The number of cells with food is M. The world time is discrete: $t = 1, 2, \ldots$ At the initial moment ($t = 1$) the food elements are randomly distributed across the cells. When $t = 1$, all prey agents are at random cells. The synapse weights of the neural networks of the prey agents are also random. It is assumed that the number of prey agents N_a does not exceed a limit: $N_a <= N_{amax}$.

Let us describe the actions of the prey agents. In each time increment, each prey agent performs one action. The actions of the prey agents are governed by their neural networks.

In a peaceful strategy, the prey agents can: (1) rest (do nothing); (2) feed; (3) move to neighboring cells; and (4) breed.

In a defense strategy, the prey agents can: (1) escape from the predator agent (if the prey agent finds the predator in the same cell, it moves to a neighboring cell); (2) threatening the predator agent; (3) attacking the predator agent (only if both the prey and predator agents are in the same cell).

Each agent has some resources (energy budget).

When a prey agent feeds on the food in its cell, the energy budget is replenished. Other actions spend the energy and reduce the agent's budget. If the energy budget goes negative, the agent dies.

With the "rest" action, the consumption of the prey agent energy is lowest.

The feeding occurs as follows. If there is food in the cell in which the prey agent is located, the agent eats that food. When the neural network orders to feed but there is no food in the prey agent's cell, the agent spends a small amount of energy identical to the "rest" action. If there is food in the cell, the prey agent eats all the available food at once. Once an agent eats food in its cell, a new food element appears in another randomly selected, food-free cell. This rule maintains the number of food elements constant.

The "move" action is moving to a neighboring cell. The direction is random.

When a prey agent breeds, the child agent appears in the same cell as the parent agent. A child is born if the total number of agents N_a is less than N_{amax}. When a new agent is born, the parent agent donates half of its energy budget to the child agent. The synapse weights of the child's neural network are equal to that of the parent's neural network with some small mutations.

Upon consuming a food element, the energy budget of the prey agent is increased by Δr_1. A prey agent's energy consumption for rest, moving to a neighboring cell, threatening the predator agent, and attacking it are Δr_2, Δr_3, Δr_4, Δr_5, and Δr_6, respectively. We assume that $\Delta r_2 < \Delta r_3 < \Delta r_4 < \Delta r_5 < \Delta r_6$.

The predator agent can: (1) rest (do nothing); (2) move to the neighboring cells to evade the threatening prey agents; (3) attack a prey agent in the same cell. If a prey agent's energy budget when attacked (and killed) by the predator goes negative, the prey agent is assumed to be eaten by the predator.

The predator's energy gain from eating a killed prey agent is ΔR_1. The predator agent's energy consumption for rest, moving to a neighboring cell, and attacking a prey agent are ΔR_2, ΔR_3, and ΔR_4, respectively. We assume that $\Delta R_2 < \Delta R_3 < \Delta R$.

Let's consider the predator agent's logic in detail. In each time increment, the predator agent performs one action as follows:

(1) The predator agent first estimates the number of threatening prey agents in its cell and the right and left neighboring cells. If the number of prey agents in the predator's cell is greater than in the neighboring cells, the predator *moves* one cell to the side where the number of threatening prey agents is smaller; if this number is the same on both sides, the predator chooses the side to move randomly. Additionally, with a certain probability P_{move} the predator can move to a cell with fewer threatening prey agents regardless of the number of the threatening agents in the predator's cell.

(2) If the predator does not evade the threatening prey agents, and there are prey agents in the predator's cell, the predator *starts fighting* the prey agents: it *attacks* a randomly selected prey agent in the predator's cell. If there are also prey agents in the cell ready to fight, they all engage in a fight against the predator. The fight reduces the energy budgets of both the prey agent (attacked by the predator) and the predator (for the energy consumption values please refer to equations (2), and (3) below). If a prey agent's energy budget goes negative (the prey agent dies), the predator agent eats it, and the predator's energy budget increases significantly. If the predator agent's energy budget goes negative, the predator dies.

(3) If the predator agent does not move away from the threatening prey agents or does not engage in a fight with them, it takes the "*rest*" action.

The predator agent's priorities are: (1) move; (2) fight; and (3) rest.

Consider the loss of the agents being attacked.

The energy loss of any agent after a hit(s) is proportional to the total strength of the hits received. For a prey agent, the loss is $\Delta r_D = k_1 F_P$, where F_P is the strength of the predator's hit; k_1 is the proportionality factor (common value for all hits). For the predator agent, the loss is $\Delta R_D = k_1 F_S$, where F_S is the total strength of all hits by the prey agents attacking the predator at the moment. The strength of an individual is assumed to be proportional to the loss of the attacker's energy. For the predator agent, the strength is $F_S = n_F k_2 \Delta r_6$, where n_F is the number of prey agents attacking the predator at the moment, and k_2 is another proportionality factor. Summarizing the above equation, we obtain that the energy loss of a prey agent when it hits the predator agent is:

$$\Delta r_D = k_1 k_2 \Delta R_4 = k \Delta R_4 , \quad (1)$$

The predator agent's energy loss is:

$$\Delta R_D = k_1 n_F k_2 \Delta r_5 = k n_F \Delta r_5 , \; k = k_1 k_2 . \quad (2)$$

That is, it is sufficient to introduce just one proportionality factor k to characterize the agents' energy losses. The number of prey agents n_F simultaneously hitting the predator agent is determined by their actions invoked by their neural network control systems.

Let us consider the *sensory signals* arriving at the inputs of the neural networks of the prey agents. These signals are:

(1) The agent's energy budget.

(2) The total number of prey agents in the nearest neighborhood of the agent (in the same cell and the two neighboring cells on the right and left; it is a single combined signal).

(3) Presence of food in the agent's cell.

(4) Presence of food in the cell on the right

(5) Presence of food in the cell on the left

(6) Presence of the predator in the agent's cell.

(7) Presence of the predator in the cell on the right

(8) Presence of the predator in the cell on the left

Therefore, there are 8 input signals and 8 inputs to the neural network of the prey agent.

Now let us describe the neural network. The outputs of the neural network control the agent's actions. The neural network has a set of synapse weights **W**. This is a single-layer artificial feed-forward neural network. To describe its operation, we will use the approach proposed in [5]. To calculate the values of the output vector **O**, the input vector I is multiplied by the weight matrix **Wk** whose values are bounded by the [$-W_{max}$; W_{max}] range:

$$O_j = \sum_i w_{ij} i_i. \quad (3)$$

The output vector **O** contains 7 components representing the following prey agent's actions:

(1) rest (do nothing)

(2) feed

(3) move to one of the neighboring cells

(4) breed

(5) escape from the predator agent by moving to a neighboring cell

(6) threaten the predator agent

(7) attack the predator agent.

At each time increment, the prey agent performs one of these actions. Usually, it is the action corresponding to the max output O_j. Besides that, with a certain probability P_{rand} a prey agent can perform another action selected randomly. More specifically, with $1 - P_{rand}$ probability the action is the one corresponding to the maximum output of the neural network, and with P_{rand} probability, the action is random. For random actions, the probabilities of selecting each of the 7 possible actions are equal. Note that P_{rand} differs for different agents and changes as the population of prey agents evolves.

The synapse weights are also adjusted in the course of evolution. The initial synapse weights of the prey agent neural networks (at $t = 1$) are random: it is assumed that the w_{ij} values are uniformly distributed in the [$-W_{max}$,$+W_{max}$] range. Once a child of a prey agent is born, it inherits the synapse weights of the parent agent's neural networks with small mutations: each weight in the parent's weight matrix is modified by adding either $-P_M$ or $+P_M$ with equal probability. The P_M value represents the rate of mutations. The synapse weights cannot be beyond the acceptable [$-W_{max}$,$+W_{max}$] range.

The probability of randomly selecting a P_{rand} varies as follows.

The initial P_{rand} values at $t = 1$ for all agents are identical: P_{rand} ($t = 1$) = P_{rand0}. Then the P_{rand} values change during breeding: they are inherited in some variations. For a child agent, a value uniformly distributed in the [$-P_r$, $+P_r$] range is added to the P_{rand} of the parent agent. The P_{rand} values cannot exceed the [0, 1] range. Note that random selection of actions is similar to noise or random evolution and optimization of the prey agent behavior. Intuitively, a higher rate of random search can be beneficial when the agent's behavior is far from optimal. Otherwise, the rate can be reduced.

We used computer simulation to analyze the model.

At the initial moment, we defined a grid world with some food in the cells. All prey agents and the predator were put into the cells. The food elements and the agents were randomly placed in the cells. Next, initial neural networks of the prey agents were built. For each prey agent, we specified the probabilities of randomly selecting the action P_{rand} ($t = 1$) = P_{rand0}. Then the agents operated as described above.

Since some of the prey agents may die from predator attacks or due to dropping the energy budget below zero, we counted the "live" agents in the population at each time increment. If the number of agents became less than the initial population size of prey agents $N_a(t = 1) = N_{a0} = 100$, we added new agents. The positions and synapse weights of these new agents were randomized. The energy budget and probability of randomly choosing the action P_{rand0} were equal to that of a prey agent in the initial population.

The control systems of the prey evolved, and the agent population self-organized. There was no training. It was a pure evolution and survival of the fittest agents.

3. Computer Simulation Results

3.1. Basic Simulation Parameters

The size of the grid world is $N = 50$ cells.

The number of cells with food is $M = 25$.

The initial population of prey agents is $N_a(t = 1) = 100$.

The max number of prey agents is $N_{amax} = 200$.

The max value of the synapse weight $W_{max} = 1$.

The mutation rate, which represents the parent-child changes in the synapse weights is $P_M = 0.03$.

The initial probability of randomly selecting an action is $P_{rand0} = 0.3$.

The variation of the parent-child probability of randomly choosing an action is $P_r = 1$.

The probability of the predator evading the threatening prey agents (see the predator agent behavior above) is $P_{move} = 0.5$.

A prey agent's energy gain after eating a food element is $\Delta r_1 = 0.1$.

A prey agent's energy loss for resting is

$\Delta r_2 = 0.005$.

A prey agent's energy loss for moving by one cell is $\Delta r_3 = 0.01$.

A prey agent's energy loss for breeding is $\Delta r_4 = 0.02$ (besides that, a parent agent gives half of its energy budget to the child).

A prey agent's energy loss for threatening the predator is $\Delta r_5 = 0.03$.

A prey agent's energy loss for hitting the predator is $\Delta r_6 = 0.05$.

The predator's energy gain when eating a killed prey agent is $\Delta R_1 = 1$.

The predator's energy loss for resting is $\Delta R_2 = 0.01$.

The predator's energy loss for moving by one cell is $\Delta R_3 = 0.02$.

The predator's agent energy loss for hitting a prey agent is $\Delta R_4 = 0.5$.

The initial energy budget of a prey agent is $r(t = 1) = 1$.

The initial energy budget of the predator agent is $R(t = 1) = 10$.

The proportionality factor k, which represents the agents' energy losses when they are hit (refer to equations (1), (2) above) is $k = 1$.

Note that some parameters were varied during the computer modeling. The results of this variation are covered separately.

3.2. Dynamics of Prey Agents Population in the Presence of a Predator. Basic Simulation

We analyzed how the following variables vary in time: the population-average energy budget of a prey agent $r(t)$, predator's energy budget $R(t)$, the number of prey agents performing any of the 7 actions, the share of actions performed by the predator at a given moment, the number of prey agents dead by the current time increment N_{ad}, the total number of prey agents, the average probability of a random selection of an action P_{rand} (t). We simulated 1,000 time increments. All the resulting time dependencies were averaged over 1,000 simulation runs.

Fig. 1 shows the population-average energy budget of the prey agents and the predator energy budget vs. time.

As can be seen from Fig. 1, after a short initial period (when $t < 200$), the energy budget of the prey agents grows steadily. The predator's energy budget first grows and then drops to zero. Initially, the prey agents often die and get eaten by the predator, so the predator's energy budget grows at first. Subsequently, the predator's energy budget decreases, and at $t = 831$ the predator dies in all 1,000 independent simulation runs.

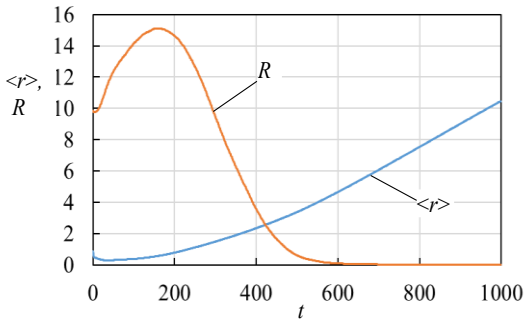


Fig. 1. Population-average energy budget of the prey agents <r> and predator's energy budget R vs. time t.

Fig. 2 shows the number of prey agents' actions vs. time for a peaceful strategy (rest, feed, move to neighboring cells, breed).

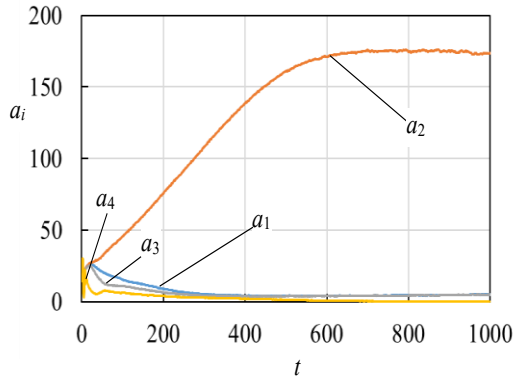


Fig. 2. The number of prey agents' actions vs. time for a peaceful strategy: rest (a_1), feed (a_2), move to neighboring cells (a_3), and breed (a_4).

It can be seen that at the very initial moments, the prey agents intensively breed and spend their energy budgets including the parent agents giving half of their budgets to children. The analysis of the computer simulation results indicates that rather quickly the agents stop having sufficient energy for breeding, and the breeding rate falls. Fig. 2 also demonstrates that the number of "feed" actions grows intensively in time and becomes predominant.

Fig. 3 shows the number of prey agents' actions vs. time for the defense strategy (evading, threatening, attacking the predator agent).

Figs. 2, and 3 show that at the initial moments, the actions of the prey agents are rather chaotic. Only after $t \approx 800$, the distribution of prey agents' actions stabilizes. Note that Figs. 2, and 3 show the agent's intended actions of feeding and attacking the predator. A real attack on the predator occurs only under certain conditions: (a) the predator must be in the same cell as the prey agent, and (b) the predator (like a prey agent) must also attack the prey agent. The "feed" action also occurs under a certain condi-

tion: when there is a food element in the cell occupied by the prey agent.

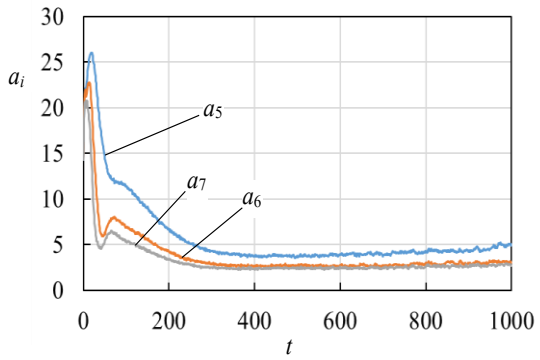


Fig. 3. The number of prey agents' actions vs. time for the defense strategy: evading (a_5), threatening (a_6), and attacking the predator (a_7).

Fig. 4 shows the share of the predator's active actions (evading threatening prey agents, hitting prey agents, resting) for 1,000 cases.

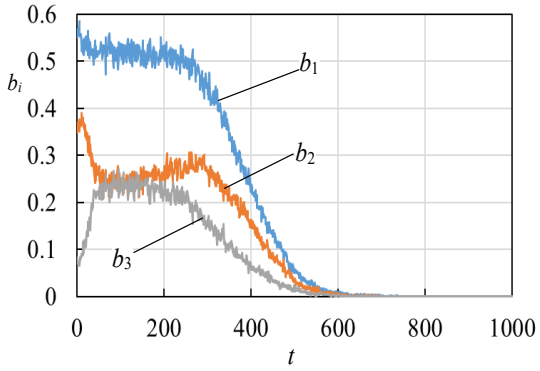


Fig. 4. The share of the active actions of the predators engaged with the prey agents vs. time t: evading threatening prey agents (b_1), attacking a prey agent (b_2), resting (b_3).

The number of deceased prey agents vs. time is shown in Fig. 5.

The total number of prey agents vs. time is shown in Fig. 6.

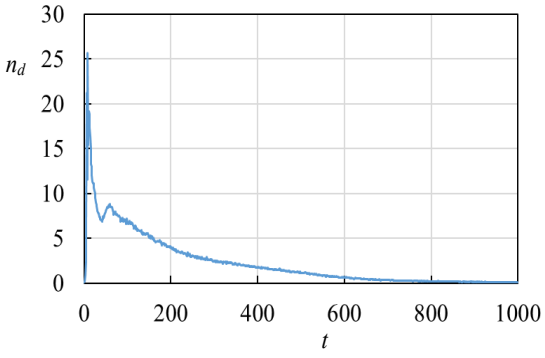


Fig. 5. The number of dead prey agents vs. time t.

Fig. 7 shows the probability of randomly selecting an action P_{rand} (t) for the prey agents vs. time. It can be seen that in the initial moments, the intensity of the random search for an action is high. As time passes, the intensity decreases. This is similar to the "gene-mutator" model [8], which assumes that the mutation rate can vary and be inherited. If a population enters a new environment, where active random search for new properties is advantageous, the mutation rate increases, while during prolonged residence in a stable environment, where preservation of already attained properties is more important, the mutation rate decreases.

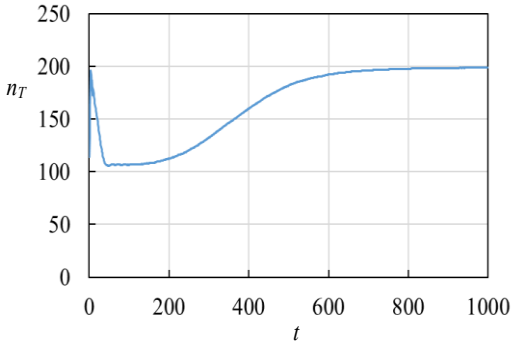


Fig. 6. The total number of prey agents n_T vs. time t.

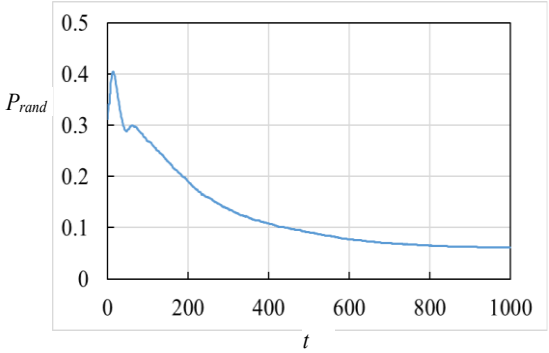


Fig. 7. The probability of randomly selecting an action P_{rand} (t) for the prey agents vs. time t.

3.3. Parameter-Varying Simulation

We ran 3,000 such simulations and averaged the results over 100 independent runs.

We varied the grid world size. For the basic simulation, the world size was $N = 50$ cells. When we reduced it to $N = 10$ cells, the dependences were similar to the ones above for $N = 50$. The significant changes for $N = 10$ are as follows: the number of the prey agents' "feed" actions decreases slightly; (2) the energy budget of the prey agents at $t = 1{,}000$ decreases significantly to $r(1{,}000) \approx 6.6$, while at $N = 50$ it is $r(1{,}000) \approx 10.5$ (refer to Fig. 1). This can be interpreted as follows. When the world is small, the prey agents have less food, so their energy budgets decrease.

When the world size increases to $N = 100$ cells, the time dependencies are similar to the $N = 50$ case. The changes are as follows: (1) for 100 independent simulation runs, in all cases the predator dies before $t = 1{,}746$; (2) the prey agents' energy budget at $t = 1{,}000$ decreases to $r(1{,}000) \approx 2.3$. We also observed that at $N = 100$, sometimes the foo elements were distributed unevenly: (a) there were cell chains (approximately 10 cells long) with no food at all; (b) there were cell chains with food in each cell. Similar results were reported in [9]. Due to such an uneven distribution, the prey agents had difficulty finding food and therefore had a radically reduced energy budget compared to the $N = 50$ case.

We also ran simulations with fewer food elements than in the main analysis ($M = 25$). We considered $M = 20$, $M = 15$, $M = 10$ ($N = 50$ for all cases). At $M = 20$, the results only change slightly: the prey agents' energy budget decreases (at $t = 1{,}000$ it is $r(1{,}000) \approx 6.4$) and the predator's lifespan changes slightly (at $t = 925$ the predator dies in all 100 independent simulations). At $M = 15$, the situation changes significantly: the prey agents' energy budget decreases significantly, and the predator agent survives up to $t = 3{,}000$. The prey agents' and predator's energy budgets for $M = 15$ are shown in Fig. 8. At $M = 10$, the variations of the variables change drastically: the predator survives until $t = 3{,}000$ in all 100 independent simulations, and with time the predator's energy budget increases to $R(3{,}000) \approx 150$, while the prey agents' energy budget decrease to $r(3{,}000) \approx 0.2$, i.e. in this case the predators actively feed on the prey agents, replenishing their energy budgets, and suppress the "life" of the prey agents.

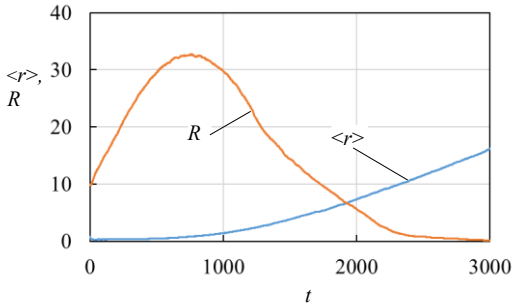


Fig. 8. Population-averaged prey agents' energy budgets <*r*> and predator energy budget *R* vs. time *t*. The number of food elements is reduced to $M = 15$.

4. Conclusion

We created and analyzed a computer simulation model representing a collective defense by weak prey agents against a strong predator agent. It is shown that a group of prey agents is able to resist a strong predator agent. In particular, prey agents can threaten and collectively attack the predator. By evading threatening prey agents, and repelling their attacks, the predator loses its energy and may die. We also demonstrated that prey agents need a fair amount of food to be able to defend themselves. Given enough food, relatively weak prey agents overwhelm a strong predator. When the prey agents' food supply is low, the predator agent suppresses the prey agents.

This is a part of the FNEF-2024-0001 Development and Deployment of Trusted AI Systems based on New Mathematical and Algorithmic Approaches and Fast Computing Models Compatible with Domestic Computer Hardware government contract (10230321,00070-3-1.2.1).

Модель обороны коллектива слабых мирных агентов от сильного агента-хищника

В.Г. Редько

НИЦ «КУРЧАТОВСКИЙ ИНСТИТУТ» — НИИСИ, г. Москва, Российская Федерация;

vgredko@gmail.com

Аннотация. Построена и изучена модель взаимодействия сообщества относительно слабых мирных агентов в клеточном мире с сильным агентом-хищником. Агент-хищник может нападать на мирных агентов, убивать и съедать их. Внутренняя система управления мирного агента представляет собой нейронную сеть. Имеются две стратегии мирных агентов: 1) обычная мирная жизнь, 2) оборона от сильного агента-хищника. В первой стратегии мирные агенты выполняют следующие действия: находиться в состоянии покоя, питаться, размножаться, перемещаться по миру. Во второй стратегии действия мирных агентов таковы: уход от агента-хищника, угроза агенту-хищнику, нападение на агента-хищника. Выходами нейронной сети являются сигналы, определяющие действия мирного агента. Агент-хищник выполнять следующие действия: находиться в состоянии покоя, уходить от угрожающих мирных агентов, нападать на мирных агентов. Повеление агента-хищника определяется простыми логическими правилами. Анализ модели производился путем компьютерного моделирования. Показано, что при достаточно естественном выборе параметров модели коллектив мирных агентов побеждает агента-хищника, а именно, с течением времени ресурс мирных агентов уверенно растёт, а ресурс

агента-хищника в итоге уменьшается до нуля, т.е. агент-хищник погибает. Также продемонстрировано, что для обеспечения способности к такой обороне, мирным агентам нужно достаточно большое количество пищи. При малом количестве пищи мирных агентов агент-хищник подавляет мирных агентов.

Ключевые слова: мирные агенты, агент-хищник, борьба между агентами

References

1. C.G. Langton (ed.). Artificial Life: The Proceedings of an Interdisciplinary Workshop on the Synthesis and Simulation of Living Systems. Redwood City CA: Addison-Wesley, 1989.

2. C.G. Langton, C. Taylor, J.D. Farmer, S. Rasmussen (eds.). Artificial Life II: Proceedings of the Second Artificial Life Workshop. Redwood City CA: Addison-Wesley, 1992.

3. L.S. Yaeger. Computational genetics, physiology, metabolism, neural systems, learning, vision, and behavior or PolyWorld: Life in a new context. In: C.G. Langton (Ed.). Proceedings of the Artificial Life III Conference. Reading, Mass.: Addison-Wesley, 1994, 263–298.

4. D. Ackley, M. Littman. Interactions between learning and evolution. In [2], 487–509.

5. M. Burtsev, P. Turchin. Evolution of cooperative strategies from first principles. “Nature”, V. 440 (2006), No. 7087, 1041–1044.

6. K. Lorenz. On Aggression. London, New York, Routledge Classics, 2002.

7. V.G. Red’ko. Model of collective interaction in competing groups of autonomous agents. In: V. Redko, D. Yudin, W. Dunin-Barkowski, B. Kryzhanovsky, Y. Tiumentsev (eds). Advances in Neural Computation, Machine Learning, and Cognitive Research VIII. NI 2024. Studies in Computational Intelligence, V. 1179, Springer, Cham, 2025, 345–352.

8. M.A. Semenov, D.A. Terkel. Об эволюции механизмов изменчивости посредством косвенного отбора. *Journal of General Biology*, Vol. 46 (1985), No. 2, pp. s271–277.

9. Red’ko V.G A Model of Interaction and Competition Between Two Types of Autonomous Agents. *Russian Journal of Cybernetics*. Vol. 6 (2025), No. 1, pp. 128–136.

Наименование: сетевой рецензируемый научный журнал «Труды НИИСИ»
Сведения о переименовании: до 2025 года журнал издавался в печатном виде с названием «Труды НИИСИ РАН», ISSN 2225-7349
Журнал основан: 2011 г.
Периодичность: 4 раза в год
Учредитель и издатель: НИЦ «Курчатовский институт» — НИИСИ
Главный редактор: Бетелин Владимир Борисович, д. ф.-м. н., профессор, академик РАН
Адрес учредителя и издателя: 117218, Москва, Нахимовский проспект, д.36, к.1
Адрес редакции: 117218, Москва, Нахимовский проспект, д.36, к.1
Контакты: Тел.: +7 (925) 924-83-46; muranov@niisi.msk.ru
Подписка: Электронная версия журнала находится в свободном доступе на сайте журнала, а также в базах данных открытого доступа

Title: *SRISA Proceedings* online peer-reviewed journal
Former title: until 2025, the journal was published in print under the name *Trudy NIISI RAN* (Proceedings of SRISA, Russian Academy of Sciences), ISSN 2225-7349
Published since: 2011
Publication frequency: Quarterly
Founder and publisher: NRC "Kurchatov Institute" - SRISA
Chief Editor: Vladimir B. Betelin, Doctor of Science (Phys&Math), Prof., member of the Russian Academy of Sciences,
Address of the founder and publisher: 117218, Moscow, Nakhimovsky avenue, 36, bldg. 1
Address of the editorial office: 117218, Moscow, Nakhimovsky avenue, 36, bldg. 1
Contacts: Tel.: +7 (925) 924-83-46; muranov@niisi.msk.ru
Subscription: The electronic version of the journal is freely available on the journal's website, as well as in open access databases

Подписано в печать 10.06.2025 г.
Формат 60x90/8
Печать цифровая. Условных печатных листов – 9,0